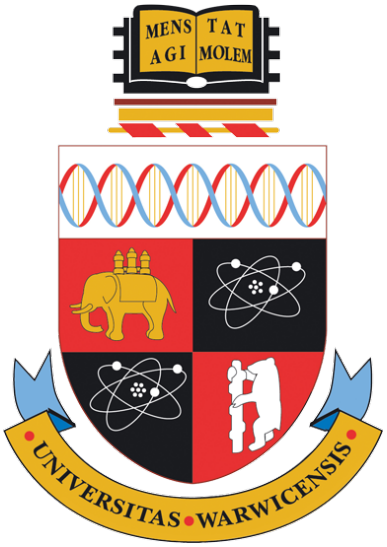
WARWICK
THE UNIVERSITY OF WARWICK



Automating SLA Enforcement in the Cloud Computing

by

Farrukh A. Qazi

Thesis

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Computer Science

December 2020

Contents

List of Tables

List of Figures

Acknowledgments

First and foremost, I humbly and exceptionally feel indebted to my creator for countless and continuous blessings as without that, it was impossible to get at this stage for seeking higher education.

I thank the EPSRC funding authorities for awarding me the scholarship for this research program through Warwick University.

Since day one, I found my supervisor, Dr. Arshad Jhumka, a very humble, inspiring and extremely passionate intellectual. Throughout my research, whenever a difficult situation arose, he always offered his compassionate support to me. While performing this academic research there are times when one's morale can go down due to frustration. However, his consistently encouraging attitude always kept me intact on this research voyage. Being an exceptional and friendly mentor, he treats everyone equally with full respect and honour.

I highly admire the marvellous people, I met here at the Department of Computer Science (DCS), The University of Warwick, whether they belong to the academic team or they are part the admin/ technical support. Everyone has been truly supportive and incredibly encouraging throughout my research. My group team members at DCS have also been very admiring, facilitating and supportive, undeniably they are all true friends.

This humble research work is being dedicated to my beloved Spiritual Teacher ***Al-Shaikh Al-Hazrat Abu Anees Muhammad Barkat Ali Ludhianvi Q.A.***
who has always been a beacon of light for me.

I am also very grateful to my family, for their excellent support, continuous encouragement and patience. They offered the best possible support to facilitate my uninterruptible research work which meant a lot to me and I will always owe them.

Declarations

This thesis is my own work and has not been submitted for a degree at another university.

Part of this thesis has been published by the author, as follows: Chapter 5 reflects the same publication.

[160] Farrukh Qazi, Arshad Jhumka, and Paul Ezhilchelvan. Towards automated enforcement of cloud sla. In *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*, pages 151–156. ACM, 2017

Abstract

Cloud computing is playing an increasingly important role, not only by facilitating digital trading platforms but also by transforming conventional services from client-server models to cloud computing. This domain has given the global economic and technological benefits, it offers to both the service providers and service subscribers. Digital marketplaces are no longer limited only to trade tangible commodities but also facilitates enormous service virtualization across various industries. Software as a Service *(SaaS)* being the largest service segment, dominates the global cloud migration. Infrastructure as a Service *(IaaS)* and cloud-based application development also known as Platform as a Service *(PaaS)* are also next-generation computing platforms for their ultimate futuristic demand by both, public and private sector. These service segments are now hosted on cloud platforms to compute, store, and network, an enormous amount of service requests, which process data incredibly fast and economically.

Organizations also perform data analytics and other similar computing amenities to manage their business without maintaining on-premise computing infrastructures which are hard to maintain. This computing capability has extensively improved the popularity and increased the demand for cloud services to an extent, that businesses worldwide are heavily migrating their computing resources to these platforms. Diverse cloud service providers take the responsibility of provisioning such cloud-based services for subscribers. In return, a certain subscription fee is charged to them periodically and depending upon the service package, availability and security. On the flip side, such intensive technology shift and outsourcing reliance have also introduced scenarios that any failure on their part leads to serious consequences to the

business community at large.

In recent years technology industry has observed critical and increased service outages at various cloud service providers(CSP) such as Amazon AWS, Microsoft, Google, which ultimately interrupts the entire supply chain and causes several well-known web services to be taken offline either due to a human error, failed change control implementation or in more recently due to targeted cyber-attacks like DDoS. These web-based solutions such as compute, storage, network or other similar services are provisioned to cloud service subscribers (CSS) platforms. Regardless of a cloud service deployment, a legal binding such as a Service Level Agreement (SLA) is signed between the CSP and CSS. The SLA holds a service scope and guarantees in case of failure. There are probabilities where these SLA may be violated, revoked, or dishonoured by either party, mostly the CSP.

An SLA violation along with an unsettled dispute leads to some financial losses for the service subscribers or perhaps cost them their business reputation. Eventually, the subscriber may request some form of compensation from the provider such as a service credit or a refund. In either case, the burden of proof lies with the subscribers, who have to capture and preserve those data or forensically sound system or service logs, supporting their claims. Most of the time, this is manually processed, which is both expensive and time-consuming. To address this problem, this research first analyses the gaps in existing arrangements. It then suggests automation of SLA enforcement within cloud environments and identifies the main properties of a solution to the problem covering various other avenues associated with the other operating environments.

This research then subsequently proposes architectures, based on the concept of fair exchange, and shows that how intelligently the approach enforces cloud SLA using various techniques. Furthermore, by extending the research scope covering two key scenarios *(a)* when participants are loss averse and *(b)* when interacting participants can act maliciously. Our proposed architectures present robust schemes by enforcing the suggested solutions which are effective, efficient, and most importantly resilient to modern-day security and privacy challenges.

The uniqueness of our research is that it does not only ensure the fairness aspect of digital trading but it also extends and logically implements a dual security layer throughout the service exchange. Using this approach protects business participants by securely automating the dispute resolutions in a more resilient fashion. It also shields their data privacy and security from diverse cyber challenges and other operational failures. These architectures are capable of imposing state-of-the-art defences through integrated secure modules along with full encryption schemes, mitigating security gaps previously not dealt with, based upon fair exchange protocols. The Protocol also accomplishes achieving service exchange scenarios either with or without dispute resolution. Finally, our proposed architectures are automated and interact with hard-coded procedures and verifications mechanism using a variant of trusted third parties and trusted authorities, which makes it difficult to cause potential disagreements and misbehaviours during a cloud-based service exchange by enforcing SLA.

Acronyms

ACL Access Control List.

ART Average Response Time.

CA Certificate Authority.

CSA Cloud Service Agreement.

CSFL Cloud Service Forensics Logs.

CSP Cloud Service Provider.

CSS Cloud Service Subscriber.

FHE Fully Homomorphic Encryption.

HTTP Hypertext Transfer Protocol.

IaaS Infrastructure as a Service.

ITSM IT Service Management.

KPI Key Performance Indicator.

OLA Operational Level Agreement.

PaaS Platform as a Service.

PHE Partially Homomorphic Encryption.

QoCS Quality of Cloud Service.

QoE Quality of Experience.

QoOSE Quality of Overall Service Experience.

QoOUE Quality of Overall User Experience.

QoS Quality of Service.

QoU Quality of Usage.

REST Representational State Transfer.

RoI Return of Investment.

SaaS Software as a Service.

SCM Secure Computing Module.

SLA Service Level Agreement.

SLC Service Level Compliance.

SLM Service Level Management.

SLO Service Level Objectives.

SMP Secure Module Platform.

SOAP Simple Object Access Protocol.

SOAR Security Orchestration, Automation, and Response.

TA Trusted Authority.

TMP Trusted Module Platform.

TTP Trusted Third Party.

UDDI Universal Description, Discovery and Integration.

UMIN Unique Message Identification Number.

WSDL Web Services Description Language.

Chapter 1

Introduction

Distributed computing is known by a metaphor called cloud computing *(CC)* through its tremendous impact on our relentlessly evolving and innovative economic surface. CC consists of multiple distributed computing nodes forming clusters of various interconnected networks along with other smart and resilient computational resources regardless of their geo-locations which empowers speedy data processing to facilitate and resolve resource hungry and complex computing problems. It combines various computing frameworks, architectures for a joint computing effort, which works like a huge computing resource to serve multiple clients, requesting diverse compute instructions through various user interfaces. The fact of their on-demand, timeless, and serviceability with least resource required to access web-based applications. Using browser-based technologies makes these computing resources more attractive, productive, and most importantly economical for its clients. CC is potentially inspired by the client/ server architecture. The basis of these architectures enables a system to be methodically distributed across diverse platforms in sundry environments. Client's service requests are processed by the server machine *(back-end system with more compute resources available)* who dispatches anticipated responses back to the client machine *(front-end system with comparatively less compute resources available)*. All these requests are generated through the client, whereas the server is obliged to process responses if all the communication terms and interaction conditions are met, within an agreed time frame.

Early days of client/server environments, where computing resources became affordable, businesses started keeping them on-site such as dedicated computer centers. Gradually enterprises started to invest by extending their *(power-hungry)* computing resources to match their business requirements as well to avail efficient data processing capabilities. The dark side to this was owning big servers and printers, which was a huge cost towards their maintenance, security. Their non-suitability for the environment was also becoming a

concentration point to the global business community. On the other side, the fact of retaining unwanted, swiftly outdated computing resources was also a big challenge. A sudden shift into the computing industry offered the idea of outsourcing these computing facilities to match client's computing demands. This infrastructure was kept remotely in commercial data centers, owned by third parties. These computing service providers offered such economical, on-demand digital resources equipped with cutting edge technologies, that clients couldn't resist, therefore, the concept of maintaining their computing facilities started diminishing within a few years. Cloud computing emerged and took the consumer market swiftly, to full fill their data processing needs in a more reliable, efficient, effective, and economical fashion.

The service subscribers are obliged to pledge the least financial and resource commitments which enable them to access state-of-the-art cloud-based resources with convenience. These technologies serve businesses such as compute, storage, network, or alike digital service provisioning which miraculously meet their business expectations and computing needs.

The concept of on-demandability, economic competitiveness, and cross products inter-portability has certainly enhanced novel concepts and competitive edges from the service providers to the service subscribers. This has also renovated and rebranded the entire supply chain, regardless of products or business domains. Within the cloud computing realm, the transitioning from on-premise to cloud computing is where enterprises are focusing on the concept of virtualization which convinces them for acquiring virtual instances of computing resources, automation processes, and interactive services to facilitate their business goals as well as their clients.

Innovative trends within the electronic commerce market and its constant transitioning have opened new horizons towards all emerging digital trading. According to Gartner the forecast of global public cloud revenue is predicted to be grown 6.3% in 2020 to a total % 257.9 billion, up from $242.7 billion in 2019,[191].

As CC was further emerging into an excellent resource because of continuous improvements with the help of researchers and industrial practitioners and also with an association of esteemed vendors, it was attracting more and more organizations both in the private and public sector to embrace these platforms. As the evolution of utility computing was being extended from old concepts such as time-sharing servers into virtualization was coming live into the commercial computing paradigm. The CC service catalog was initially introduced to facilitate some limited type of computing aspects, however, as time passed these trends get changed and multiple service flavors were being made available. This includes cloud service models (public cloud, private cloud,

hybrid clouds) and deployments models (Software as a Service (SaaS), Platform as a service (PaaS), Infrastructure as a service (IaaS)).

Every organization holds the projection of their potential cloud-based usage so they can concentrate only on their business processes through these cloud services. Most of the cloud-based resources are calculated on *pay-as-you-go* basis. Their flexible and on-demand instance deployment from a cloud service provider *(CSP)* would encourage cloud service subscribers *(CSS)* to only emphasize specific services or resources. The prevents them to avoid any unnecessary financial overheads such as unanticipated billing costs for unused CPU cycles, storage locations, or allocated memory blocks. Once the required cloud services are provisioned, clients evaluate the performance of those web services through their projected key metrics such as response time, uptime, availability, and other vital Key performance indicators *(KPIs)* stating the web application suitability for the purpose it's being provisioned. The CSP measures and publishes those strategic stats such as numbers of 9s. Three 9s means a service guarantee is pledged by the CSP that the service will be up 99.9% or four 9s represents a 99.99% availability [135]. Provisioning highly available and fail-safe resources incur extra cost to the CSS, if they don't need such availability until they intend to run business-critical web services such as health, aviation, stock markets, or other alike services, they do not sign-up for such resources.

CSP being an outsourcing entity, offers their infrastructure commercially, which of course increases their sales. Offering such shared services on multi-tenancy business models, where data is processed, stored, and transmitted through CSP's owned infrastructure to their multiple clients. Either a human error, system misconfiguration or an attempt launched by some unsolicited attacker(s) *(internal, external)* could introduce serious security threats. It becomes a perilous issue not only for CSP, the owner, and maintainer of that facility but the CSS whose data, reputation would be at stake if they have to face a media or a legal trial for getting involved with a data security/privacy compromise. To avoid such challenges and to get a smooth service delivery, both the CSP and the CSS get engaged through a signed contract e.g. Service Level Agreement *(SLA)*. Such initiative enlists key facts about provisioned cloud service, its scope, costing, etc., however, due to above-mentioned vulnerabilities and in the presence of uncertain hostile actors and adversaries, the CSP may cease the cloud services, cause unsolicited delays, stagger other unpredicted and intermittent service interrupts. This becomes unacceptable for the CSS as they might have paid for these services upfront to the CSP. Exercising their legal or regulatory obligation, a cloud service consumer can escalate such matters to the legal authorities to prosecute the service provider in

terms of demanding reasonable compensations, for breaching the contract, and to cover their(CSS's) *(financial/ reputational)*losses. This refers to instances when CSS has to experience unscheduled service outages, disruptions during the agreed service term. There could be a situation when the CSP uses the CSS for some non-payments. Due to all these circumstances, a well-structured and appropriate infrastructure is needed, which can flawlessly manage unbiased arbitrations. A fully integrated system, which can instantly resolve disputes while confronting other associated properties for such negotiations. Where fundamentals obligation like security is unconditionally ensured and guaranteed to a good acceptable and confident level.

There is a problem called, *fair exchange* that stipulates vital support towards this challenge. It also embraces those critical properties such as non-repudiation, timeliness, and security. My research would check the suitability of fair exchange, as a building block for SLA enforcement within cloud computing environments in the presence of diverse attackers and other adversaries. My thesis is about the fact, that CSS and CSP contractually confined through SLAs, so is it possible that fair exchange protocol *(FEP)* is the perfect solution? At these preliminary stages, I am trying to plan a framework using FEP for this purpose.

1.0.1 Motivation

Enterprises are already making huge efforts to expeditiously migrate their office automation and production processes to cloud computing environments. These cloud-based services extensively resolve a lot of problems such as having on-premise infrastructure for their day-to-day data processing requirements. To facilitate a confident and trustworthy technology shift, both the CSP and the CSS sing an SLA which binds them for the corresponding obligations and segregation of contractual services. SLA violations do occur during the service provisioning because of various issues where either the service provider or in some cases the service subscriber deviate from their commitments as defined in the SLA. Holding someone responsible such as in the case of CSS who has to suffer from unscheduled service outages or perhaps the if QoS doesn't meet their business requirement. Similarly, if the CSP gets into a situation where the CSS doesn't pay their dues, in either case. there will be a party who will be victimized by losing them monetarily or even their business reputation. Moreover, dealing with SLA violations does require manual work such as the CSS would lodge a report stating what violation has been observed. CSP investigations cost the CSS and also reduced productivity however, the most intolerant part is when CSP demands the CSS for some satisfactory proof

justifying those SLA violations. This could lead to a situation where such SLA violations could mean some metadata that is classed as forensically sound digital evidence. This research was motivated to investigate such scenarios where dispute resolution can be robustly managed and the perpetrator can be held liable for such intense commercial damages. Fair Exchange protocols have been discussed and framed to work with diverse platforms. This research goes a step ahead by evaluating FEP capabilities and suitability for SLA enforcements smartly. My prime curiosity behind this entire research work was based upon the following research questions which were outlined and kept me motivated to seek some valuable results:-

Question-1 *Is it possible to use fair exchange as base protocol for cloud SLA enforcement?*

Question-2 *What would be the most appropriate protocol to work with when service exchange participants are loss averse?*

Question-3 *How a protocol will mitigate various threats if participating entities act maliciously while the service exchange is being performed in cloud environments?*

1.0.2 Thesis Contribution

The problem of fair exchange is to ensure that when two entities wish to exchange their respective items *(e.g. data, service, financial)*, no one entity has an unfair advantage over the other. This research work thus makes the following contributions:

- *Introduces and formalizes the problem of SLA enforcement as a fair exchange problem.*
- *Proposes a protocol, based on fair exchange, that solves the SLA enforcement problem.*
- *Investigates scenarios and their implications when participants are (i)loss averse and (ii) when participants act maliciously.*

1.0.3 Thesis Structure

In this chapter, a basic introduction stating an overview of the research was presented along with the research motivation, thesis contribution. Following is the brief structure for the rest of this thesis, as a quick reference:

Chapter 2 Runs an analysis of the volume of enterprises continuously adapting cloud technologies where there is still a lack of regulatory frameworks, technical solutions to protect contractual obligations. To cope with the huge migrating community how SLA enforcement is deal with so far. It also aims to explore the process automation supporting SLA enforcement. Explores previous studies to review and correlate how researchers and practitioners have explored various options.

Chapter 3 This chapter studies fair exchange protocols and assesses their types, properties, and other associated fringe attributes in terms of their deployments and application. It also observes protocol's potential implementation, operating, technical and security requirements with convergence to SLA enforcement in service-oriented architectures.

Chapter 4 Presents two scenarios (i) a system model where two entities can perform their service exchange under some set business terms and operating conditions using diverse tools, techniques, and procedures (ii) secondly, it discusses fault models when the same participants would experience certain *(internal/external)*adversaries and analyses their *(machine, human)* behaviors to highlight potential threats.

Chapter 5 Justifies a narrative, which was worked out and published based on Fair Exchange Protocol. It explains the proposed protocol and how it works when two participants are interacting, while their intent is as loss averse. The protocol also includes a trusted third party *(TTP)* which governs the entire exchange and ensures to uphold the fairness element till the exchange concludes successfully.

Chapter 6 This chapter discovers and analyses, an extended operating diversity when either of the participant or perhaps both, decide to act maliciously instead of being loss averse, to seek unfair gains over the other, how this protocol variance shields such scenarios. The protocol also demonstrate its defenses to other delicate treats such as *security* and *privacy* so it ensures its trustworthiness and transparencies capabilities when enforcing cloud SLAs.

Chapter 7 Arguments about the protocol implementations, how various properties such as correctness, non-repudiations, work and maintained. In this chapter, the discussion about components implementation when the protocols are being set up in real-world environments has also been studied. Critical challenges, complexities especially those implications concerned to comply with regulatory and legal frameworks is also briefly touched.

Chapter 8 Summarizes the thesis by giving a quick recap of the work that has been done so far on this research voyage. It also extends and marks future tasks such as developing a framework in the real environments within the cloud computing paradigm. The work would also explore how these imminent technologies would respond when diverse cloud environments will be integrated to perform some common tasks.

Chapter 2

Related Work

Modern-day innovative marketplaces have successfully replaced legacy shopping floors with virtual markets. This transition has also changed other day to day tasks, service management, and office automation in a way that service or trading hosts prefer to lease computing resources such as compute, storage, or network which can serve both commodity traders so they can publish their products online using digital marketing catalogues, similarly, the buyers can browse through their digital items using browser technologies to perform online shopping or to perform some computing tasks such as booking appointments, paying their bills, desktop publishing, emails, etc. This mutual understanding brings both the Cloud Service Providers(CSP) who offer their computing infrastructure where the Cloud Service Subscriber(CSS) can either host or use outsourced web applications and other portal facilities to accomplish their business requirements. With the huge involvement of utility computing within e-commerce, ensuring fairness especially when both the CSS and CSP are anonymous, the end-user or service recipient's expectations are high that they will receive their items for the money they have spent. Honesty becomes inevitable for both the online buyer(service recipient) and the seller(service executor or the service host) to coordinate well to safeguard service or product delivery through contracts.

Generally, a CSP does hold a higher responsibility to ensure the reliability and quality of their cloud-based service provisioning, as these services are hosted and run on the consistent infrastructure they own. These generic services are provisioned, depending upon diverse cloud deployment and service models, which are offered to either a single business entity or multiple individual cloud tenants through the hosting facility on varying distributed service platforms. Due to various technical, business, logical reasons, service delivery can get either interrupted, fail, or stopped *(intentionally or unintentionally)*. The outcome of these outages affects service recipients who wouldn't get their item delivered

as per the subscriber's expectations promptly. In a worst-case scenario, CSP's customers might experience a total service collapse, for innumerable reasons. A CSS is an entity, who pays for some cloud-based service acquisition, in terms to have access to a cloud resource so to achieve their business goals by performing some tasks, processes, automation for a certain amount of time or a service term. This is CSS's ultimate obligation to ensure their Return of Investment *(RoI)* because they are spending heavy investments while acquiring these cloud services from an intended CSP who specializes in delivering some specific web resources.

The way to ensure the service quality and the RoI is fully achieved as expected, a contractual legal document stating the business requirements, service scope, and each participant's obligations are defined. Service Level Agreement *(SLA)* is a logical way to bind both the service provider and the service subscriber, in case of any dispute or misbehave is observed and in those circumstances, either of the participants can be held accountable and the victimized can be compensated accordingly. Within the CC service delivery environments, business entities sign SLAs before service commencement however, there are a lot of technical and logical lacunas where either party can try to cheat the other participants. The thesis discusses various problems and challenges that make the service delivery very harsh and demonstrates how CSP dominates the service delivery life cycle. Various protection schemes have been introduced so far however, the critical part of the service delivery which ensures a true sense of fairness is pretty vague. Previous work by other researchers mainly spins around the detection of SLA violations and doesn't impose such restrictions where either participant can not misbehave and cheat the others.

This research work connects those dots precisely by highlighting specific gaps, challenges, limitations concerning data privacy, security that compromises service delivery and undetected SLA violation. It discusses further associated protection elements comparing threat actors from end-to-end service delivery perspective.

In the following sections, we will analyze those elements which form and play vital roles during a web service delivery. Principally, the web service paradigm works the same whether the intended clients belong to end consumers online shopping community or the belong to the corporate end-users who are working on some generic or tailor-made online applications to perform their everyday work-related tasks.

2.0.1 Web-Based Services

Marvels of the world wide web have introduced some fantastic innovations towards facilitating both the selling and buying entities. Using various internet or intranet-based protocols web-based services have been digitized and delivered remotely to the end-users through these avenues. A web service works as the prime medium which connects the service consumers(buyers) to the service providers or the sellers, regardless of their geographic location, opposite to the monolithic computing. Web services are mainly based on the most advanced form of client-server computing model where web applications are hosted to serve various online functionaries for their intended remote clients. Web service refers to online hosted services using Extensible Markup Language (XML), Simple Object Access Protocol *(SOAP)*, Representational State Transfer *(REST)* with the conjunction of various open standards and protocols, assist users to communicate and interact via diverse communication channels. [35, 40, 143] Web Services Description Language *(WSDL)* is mainly used as a programming language that describes user interfaces, network services whereas Universal Description, Discovery and Integration *(UDDI)* is also a web service which used as a directory mechanism to discover business partners.

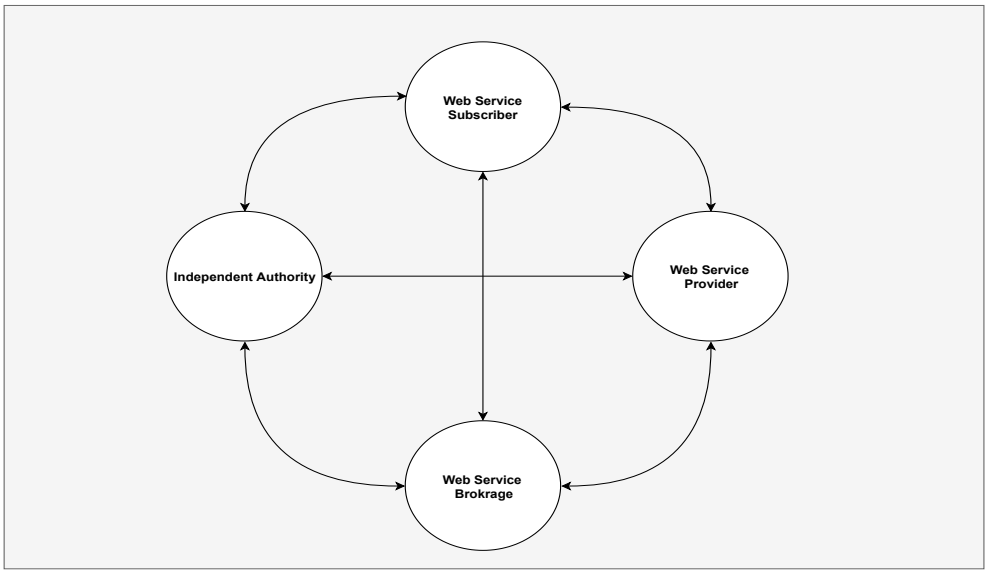


Figure 2.1: Web Service Basic Architecture

These functionalities are designed based on some pre-defined business logic empowered with innovative technologies such as *database servers, file servers, web servers, etc.* which process client requests in a fully collaborative and methodical fashion. A web service could perform various tasks such as processing an email, office automation, online information processing, etc. depending upon business requirements. A web-based service is a combination of digital resources packaged and defined logically along with their corresponding

resource identifiers made accessible through communicating protocols. An online service provider may host a site directly or on behalf of one of their client, which is packaged with various instructions pointing to some addresses and resources for further processing. Using communication protocols, web agents (browsing technologies), intended addresses, referenced web resources are some of the common components. The user interacts by sending some requests through the service provider's UI, their requests are processed by the hosting server.

After time stamping these requests, their appropriate responses for instance (HTTP response codes) service responses for being successful, redirected, failed (either client/server-side) are generated. These tasks are also logged into a repository and are eventually sent back to the initiating remote user by the hosting server if all the conditions are met. Figure 2.1 narrates a basic web service architecture which could include four entities such as service provider, subscriber, brokerage service, and some independent entity who could serve either for monitoring purposes or a guarantor depending upon the web service semantics.

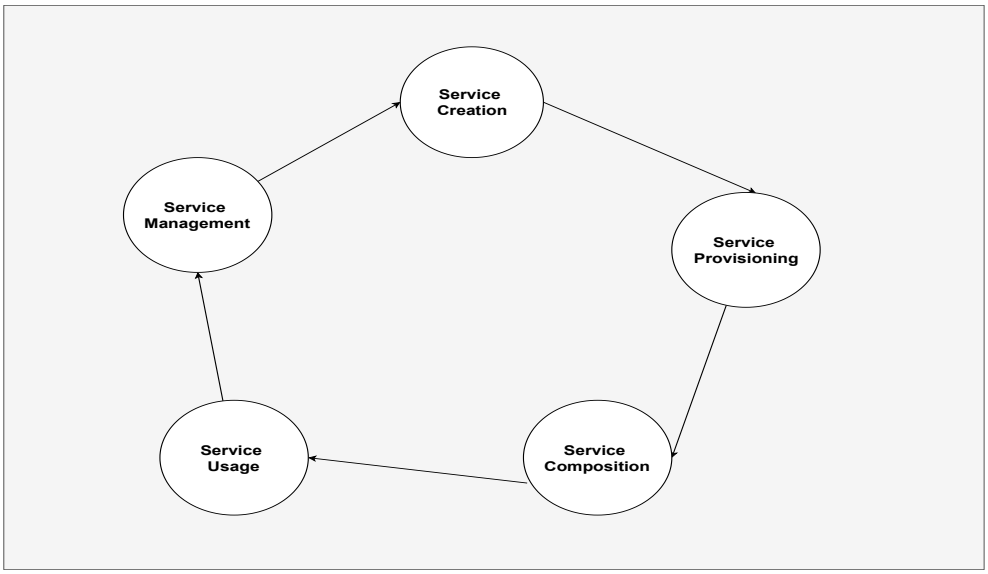


Figure 2.2: Web Service Life Cycle

A web service life cycle initiates from its creation which holds service interfaces and operation descriptions followed by other key information like the implementation plane. Service provisioning is a phase where resource allocation is defined and configured matching the client's business requirements for this service accessibility. Service Composition logically correlates to other corresponding web services or dependencies to fulfill a task in terms of its completeness. Service usage is the focal point where user requests are received, processed, and monitored. Finally, the service management is the phase that

is responsible for recording the metadata of individual service segments for future analysis which is fed through to the performance monitoring modules to detect and correlate any significant service anomalies caused by the SLA violation [177, 185].

2.0.2 Cloud Service Security & SLA Management

As more and more businesses are embracing cloud-based services, to increase their overall production. Where it increases their gross revenue by minimizing computing overheads, the magnitude of associated risks is also growing if the quality of those provisioned services does not meet the client's expectations and risk factors are not mitigated. SLA being the prime instrument binds the service providers and subscribers by defining QoS and assurance in case SLA projection fails. Cloud service providers, service brokers, and subscribers agree on SLA management procedures. SLA being a service controller ensuring service maintainability within a service-oriented production environment. The entire service provisioning is lead by CSP's defined processes, which becomes clumsy for their clients and restricts their low-level service visibility. SLA violation conditions are not fully described and intentionally written in a technically vague fashion to avoid any serious detections and potential penalties. For instance a SaaS service provisioning, the CSS won't get any low-level service metrics, seriously impacting the QoS. CSS is less privileged by not having fundamental visibility of SLA violation, intermittent outages, and informational or low urgency security incidents either. CSS might get some modified version of SLA monitoring stats, which they cannot verify or validated. This becomes a major concern to the CSS to survive with such service provisioning as their total online business reliance is on CSP's produced services. On the other hand, a CSP can also face a similar situation where a malicious subscriber can violate the SLA in certain ways.

Service Metrics and Key Performance Indicators *(KPIs)*

A metric is a quantitative measurement that refers to some events indicating system availability or correlating service uptime such as when the intended service was available with an acceptable usage along with its defined functionalities. Metric is also refereed as a ratio of time when service component is available and functional to complete some tasks and later expressed as in percentage (eg.90%) [87]. Using these numbers a cloud service can be analyzed in terms of its Quality of Cloud Service *(QoCS)*, Quality of User Experience *(QoOUE)* and Quality of Overall Service Experience *(QoOSE)* either at the end of each service term or middle of a service provisioning to calculate an average and predict the future service streaming [131, 152]. The very same metadata

also provides a service failure trail, which supplements Cloud Service Forensics Logs *(CSFL)* supporting an investigation by either the TTP, legal authority, or even a law enforcement agency. Such logs can prove service outages, process/ feature anomalies to justify whether the claimed service credits against SLA violation are legit and well justified.

Table 2.1: Service Level Agreements Classification, [39]

SLA Monitoring Factors			
Infrastructure Metrics	**SLA Elements**	**Application Metrics**	**SLA Elements**
Hardware availability	99% uptime pcm	Service-level parameter metric	Web site response time, Latency of web/db server
Power availability	99.99% pcm	Function	Average WS latency,
DC NW availability	99.99% pcm	Measurement directive	DB latency available via web resources
Backbone NW availability	99.999% pcm	Service-level objective	Service assurance
Service Credits (CS) for unavailability	Prorated on downtime	Various	Hard to predict bits
Outage notifications guarantee	Advance notice within 1 hr of complete downtime	Penalty	website/ web resource latency
Internet latency guarantee	Measured at 5-min intervals, average doesn't exceed 60ms	-	-
Packet loss guarantee	Shall not exceed 1% pcm	-	-

A service Key Performance Indicators(KPIs) are mainly measured against some predefined targets. Both the service KPIs and service metrics are part of a cloud service performance management and are the prime resources to keep track and compare the service from those expectations defined within the SLA against the actual service delivery to the client site within a service term. Most of the time a service metric cannot be categorized as a KPI due to its varying attributes. A service monitoring cannot be measured and prove its suitability without gathering precise metrics from cloud service processing nodes to user-facing UIs, as these units are fed into periodic service reports or summaries, showing service trends, effectiveness and availability so to justify

the service RoI.

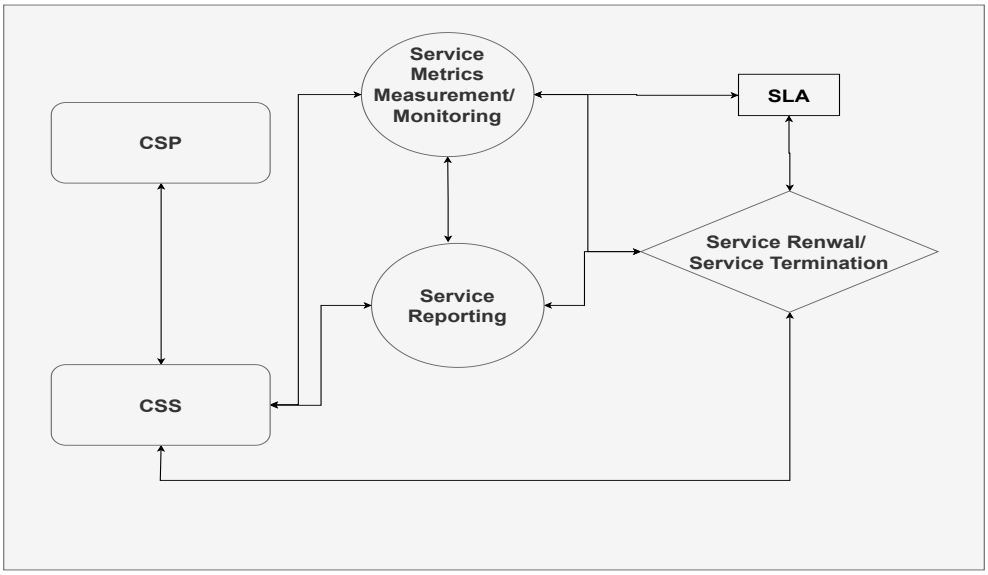


Figure 2.3: An ideal Service Monitoring state

Service metrics could be used for various purposes however, these statistics are mainly used for service agreements and for service management purposes to determine certain attributes such as service availability, reliability, response time, security, throughput, capacity, scalability, latency, supportability, economic factors. These elements assist a cloud SLA monitoring which relies on their composition, categorization, and monitoring priorities. Fig 2.3 indicates service provisioning to the CSS by the CSP, whose service requirements are measured and matched followed by periodic reports whether the SLA is met or not so the subscriber can decide if a service will be renewed or will be terminated because service requirements didn't match with the service capability or some of the above metric elements [61].

Cloud SLAs & Service Implications

Cloud computing *(CC)* is vital as the prime service originator & delivery medium. CC is made of delicate network nodes comprising highly scalable, elastic, on-demand, based inter-connected utility computing resources. These digital apparatus are fully capable to transform business requirements to compute, network, storage, and other bespoke services commercially such as Software as a Service *(SaaS)*, Platform as a Service *(PaaS)* & Infrastructure as a Service *(IaaS)*. [30, 138], are some base-line service models.
Once these cloud services are provisioned against some monetary pledges, the SLA compliance service delivery becomes the top priority for both CSP and CSS. SLA prime components are some measurable service values/metrics, service functionality when SLA could be associated with a specific service model.

SLA being the prime instrument enables both the prime parties *(CSS & CSP)* to expect their own business gains. Because SLA holds service guarantees, service metrics, service level objectives *(SLOs)* and other business, legal and technical obligations, which binds both service exchange participants into a contract. Service quality, integrity, and end-to-end performance can only be measured and monitored using some sophisticated techniques which can constitute admirable trust and a sense of fairness between these two ends. The scope of a cloud service agreement *(CSA)* [58] is also designed to benchmark all the service fragments e.g. financial, legal, operational, technical implications.

CSP focuses and expects that their digital services and automation products are provisioned across the globe so they can increase their earnings which ultimately enhance their business growth and market reputation. Ideally, the fair service accountability and consistent resource provisioning where the quality of service *(QoS)* is thoroughly monitored, should be the principal obligation of the CSP, as being an ultimate custodian, service originator, and maintainer of the underlying cloud infrastructure.

On the other hand, the CSS makes a huge financial commitment for these cloud-based services to automate business processes, production or to serve other business elements. They can perhaps also intend to resell CSP's services, acting as a cloud brokerage or a middle entity by leasing cloud services on behalf of their clients. CSS, as being the service recipient also holds a shared responsibility to measure and justify the QoS, reflecting the business-centric SLOs and performance metrics indicating the service trends, their associated penalty clauses based upon SLA, until the end of the service contract [60, 146, 153, 167].

CSP is obliged to ensure service availability, consistency, and quality until the end of the agreed term. It's also responsible for ensuring unscheduled service outages, intermittent service interruptions, privacy, security, and other similar obligations. Service guarantee indicates within a specific period how successfully the CSP can meet the SLA conditions and delivers their cloud services. For instance, a cloud subscriber intending to host a critical service such as health, financial, or another similar service, can demand at least three nines e.g. 99.9% whereas another mission-critical service subscriber could even seek a higher number of those nines e.g. *(four nines)* 99.99% or even *(five nines)* 99.999% aiming fully resilient and granular services [66, 135, 189, 212]. CSS, naturally, holds higher expectations in terms of service quality, performance, disaster recovery, and some other log management and security elements that could facilitate forensic investigations for their cloud platform or their extended client's environments [29, 153, 208].

There are so many variances and log granularities when it comes to measuring cloud services through SLAs. A service's scope, technical implication, legality binding, security, and other avenues addressing service collaboration in case of multi-tenancy could encompass various dimensions of an SLA such as facilities, platform, operating system, application, infrastructure on a higher level of visibility ensuring service performance and service quality and frequency of SLA violations [66]. Unfortunately, this course of action is not fully adopted hence leaving the CSS are made victimized by most of the CSPs. Another approach highlights the fact that mostly, the proof of burden showing SLA violations is expected from CSS, who even don't have access to the required infrastructure or operating environments where the said logs are stored [30].

SLA violation

If CSS acquires a business-critical cloud service and hosts through a leading cloud service provider (CSP) using Software as a Service (SaaS) platform. CSS would evaluate if their business requirements are meeting through this service against their RoI eg the pay-as-you-go service fees, service subscription to the CSP for some agreed amount of time. CSS's next service renewal is conditional if there is no service level agreement *(SLA)* violation occurs, otherwise, a service credit will be awarded to the CSS, upon filing their dispute for a refund. The non-compliance SLA is classed as an SLA violation by either party, however, it mainly occurs by the CSP. In another scenario, let assume, after few months of the service renewal, the CSS again observes that the QoS has been significantly reduced and SLA gets breached x-number of times. Upon escalation of the QoS, the CSS decides to file another claim to get service credits from CSP as compensation, the CSS is challenged to bring some evidence to prove their claim that CSP was responsible for SLA violation. Now when the CSS does not own the hosting infrastructure and does not have sufficient privileges to the hosting facilities so they can place appropriate monitoring arrangements in place as they were reliant on CSPs stats or SLA periodic summaries, therefore, the CSS technically would not be able to produce such data or service stats, which could add significant weight to their claims, logically, the CSP would decide to decline the claim. Due to several unsettled and disputed SLA violations, CSS potentially would lose not only their business obligations, financial losses but their reputation is also put at stack.

This makes serious business turbulence when one business entity can cheat the other and deprive them of their anticipated technical service ownerships and financial rights. SLA violation certainly costs a fortune globally when the majority of the service recipients do not know how to regain their rights

and held the cheating participants accountable for their actions. This could have been avoided if an intelligent yet automated solution were deployed to govern such service exchanges ensuring the sense of fairness at both sides [118, 142, 146]. CSP could have various motivations to violate the SLA. This could either be an intentional or unintentional reason which deprives the CSS have their due service delivery as what they have been agreed in the SLA. CSS can end up either on having less allocated technical resources such as CPU cycles, memory, storage, incensing obligation (scope, features), number of users, number of usages, number of virtual machines, average response time, latency, etc showing insufficient processing capability then anticipated, which compromises a service and flags a SLA violation occurrence. An extreme situation can also arise when a maliciously acting CSP can even terminate CSS provisioned services due to some unsolicited reason before the end of the actual service term.

SLA Verification

Cloud computing-based services and solutions certainly empower and automate phenomenal business productions by provisioning of either individual services or a group combination of interconnected services. Once these services are provisioned, keeping an eye on these services against the SLA becomes exceptionally tough as the service recipient's visibility to the factual service metrics is very limited. They are dependent on CSP statistics or SLA-related metrics to evaluate whether the services are meeting the SLA or not. In such cases, some sort of SLA verification can assist service subscribers, such that they can compare the SLOs against the real service delivery using each service element along with verification of their associated properties whether functional or non-functional [77].

SLA verification becomes more challenging when CSS and CSP are exchanging their messages over an untrusted or semi-trusted communication channel. A CSP could use these untrusted communication mediums as a justification when they intend to misbehave for instance by reducing resource allocation allowance to benefit another valued customer [211]. Verifying these services becomes an ultimate business requirement for the service subscribers so they can have a good understanding of how they have been treated by their service providers. There are a lot of schemes that have been introduced to implement such SLA service verifications[77, 106, 207, 211] on run-time, however, again a service provider with bad intention could also tackle such verification using certain techniques.

SLA Monitoring, Detection & Prevention

Service monitoring in a way that any anomalous stats, could raise a flag has become a mandatory requirement, however, such a privilege is not extended to the CSS cloud service platforms by the CSP. Monitoring schemes are mainly implemented on CSP's service delivery platforms, which mainly serves them rather than the CSS. Service monitoring means such technical implementation using bespoke software to analyze service availability, performance, or any other service dimension which comes under service disruption or even delivering a poor QoS to the cloud customers. Certain data and service points are marked using a diverse set of tools, techniques, and procedures. The entire implementation covers the entire service level management *(SLM)* feed by various factors. This may include feeds from service level compliance *(SLC)*, service-level monitoring, service level administration, which can be forwarded into a reporting node to generate periodic reports or SLA summaries for individual clients or a pool of clients in a multi-tenancy environment.

Detection of SLA violations using a proactive approach can be beneficial for both the service subscriber and the service provider, in case either of them tries to cheat their fellow business participant(s). Unsettled SLA violations cost a fortune to both, the CSP, when they have to repay financial compensations e.g. service credits against SLA violations, and the CSS when guarantees of their provisioned cloud services are compromised or failed by the CSP. Worst case scenario, service providers do disregard a CSS's legit claims. For instance, study [11, 180] shows how Amazon's S3 will class a service failure when the service availability gets below then 99.9%. Similarly, Microsoft Azure [132], Google [83], Rackspace [162] and Oracle —- have also mentioned their criteria for classifying a service outage and calculating service failure liabilities in terms of SLA violation.

Furthermore, unforeseen factors e.g. insider and outsider cyber threats could also pose serious threats to the CSS's production environment. CSP could alone act maliciously by extending resource allocation from a designated client to another resource-hungry service subscriber by stealing them quietly. They can also lower the QoS by service reduction, modification and even dropping them to significantly low to make other customers happy [211]. Regardless of such act would certainly lead to a zealous service outage [55, 89, 171, 195]. With regards to monitoring arrangements [118], logically it becomes the liability to both the parties to ensure appropriate monitoring mechanism in place however most of the CSS don't have such privileges, and hence they entirely reliant on CSP's scripts, logs, and reports. Many supporting schemes have been studied so far to facilitate such monitoring requirements can be implemented [42, 70, 137, 181,

216] however, these schemes do introduce some challenges addressing to their implementation, technical and legal aspects. These challenges do introduce various compromises throughout the entire service execution from evaluation, operational, security covering *(CIA)* Confidentiality, Integrity, Availability, privacy perspectives.

Complexities while considering the most appropriate solution which works well not only for SLA monitoring, detection and management but also integrates well with the novel cloud and IoT technologies. Earlier work shows where gathering SLA metrics for this purpose with some precise and unambiguous service specifications and the fact that these measurements carried out by the CSS are not easy, acknowledgeable, and necessarily facilitated by their service providers, the work mainly ends up on monitoring and reporting for SLA violations [179]. Similarly,[139] proposes an architecture that again monitors the SLA violation by marking various elements such as service points of presence and by involving third parties as unilaterally and bilaterally, however, enforcement is yet a challenge, in case of a violation is detected. Other previous works such as [47, 109, 114, 165, 176] are mainly aimed towards monitoring SLA violation, their dependencies, and workflow. The enforcement elements were mainly discussed by [71, 136, 151] has suggested some interesting work, where SLA monitoring and enforcement is proposed. For instance, the work done by [151] gives an impression of the complexity and limitations to implementing the enforcement on both the CSP and the CSS sides. Another piece of research was nicely performed by [202] who highlights the lack of decision making by monitoring management platforms and other similar regimes, therefore, their work spins around detection and prediction states are fed to machine learning modules which only raise alarms highlighting the SLA violations. The research work does urge for some integrated solutions which are more effective towards SLA enforcement in the modern fast pace cloud service realm.

Digital Contract Signing & Cloud Service Delivery

A contract refers to a deal where two or more business participants intend to get involved in exchanging some product or service reciprocally. They agree on declaring either one of them would become a service supplier*(seller)*, other would be classified as service recipient*(buyer)* who will be at the purchasing end against some financial commitment or even for service against a service or item against an item of mutual interest. Once they pass their initial economic negotiations, terms of services, and terms of reference are agreed upon, the service provider gets the obligation to deliver the service or service specification been agreed at the time of their negotiations. Similarly, the service subscriber also agrees in the contract that a sum of the agreed amount or something

equivalent will be paid (mostly) or delivered (instead of the agreed payment if say so in the contract) to the service provider so they can reach to the end of the contract without having any potential disputes or arguments. Any unsettled dispute might then get escalated to one of the agreed platforms such as arbitrator, negotiator, mediator, law enforcement, or even a court of law. This entire trading holds one prime aspect and that is fairness. The signed contract between all the parties involved does describe everything about their obligations, renegotiations, dispute resolutions and other do's and don'ts to avoid any disagreements while the trading is being performed or even post exchanging their goods.

Contract signing protocols are then defined as when multiple parties are involved in an exchange, they mutually follow a protocol or a set of protocols, to put a methodical function in place which could maintain maximum fairness. As per [7, 99, 112, 127] this arrangement facilitate a secure signature exchange in such a way that none of the participants could alone obtain other's digital contents unless the remaining parties have also received them without failing factors *(FF)* such as delay, modification, completeness or incorrectness.

For instance, A, B, and C are three participants for signing a contract, now none of them would be capable to receive the said contract before the other two parties do so with the notion of the above properties. The fact when either of the business participants misbehaves and does not honor business promises, the other party certainly losses their fairness which qualifies this action as a breach or violation of a contract. Using these notions, we refer to the contract signing in terms of those scenarios when a service provider who deals with some digital services advertises them on the internet, which attracts those customers who are quite keen on purchasing those items. The online service provider sets out some terms and conditions for this online sale process which offers some protection to him and in some terms protects the buyer at the same time. In case of a disagreement those terms and conditions or service agreements are reviewed and judged so if the buyer has breached those service terms, the seller will not be honoring any refunds and vice versa. However, if the service terms do protect the buyer and their trading does show that the service provider did not behave as anticipated, then it's obvious that the seller must pay the buyer their due compensation either a full refund or a service credit which can later be redeemed.

SLA Enforcement using other techniques

Enormous work has been performed in terms of managing cloud SLAs for their monitoring, guarantees, violation detection, enabling their security features

using various techniques and procedures. This research aimed to discover those avenues which have not yet been used in terms of SLA enforcement. For instance, [91] puts various filters for stripe-based and distributed-based monitoring schemes for SLA violation detection, where misbehaving flow is systematically evaluated. [111] suggested reactive and passive monitoring for this purpose stating various pros and cons for online and offline monitoring which is based on contract signing protocol to tackle various challenges. [16, 17] urges the SLA renegotiation in case of duplicated SLA violation, which in their vision not only settles the SLA implications but it also prevents over-provisioned resource management for an optimum deployment.

With regards to SLA enforcement, [151] understands the need for SLA automation, therefore, suggests a solution to this problem using logical formalization e.g. Horn Login and Event Calculus, Deontic Logic, etc. targeting complex contract rules for incorporating various conditions such as user and contractual obligation to manage and enforce individual processes. [149] extends it by reviewing SLA within an SOA architecture as containers of the functional and non-functional properties through integrating a flexible SLA enforcement layer with CSP service infrastructure. This approach rather presents an extent of complexity and introduces further challenges when enforcing a mutually agreed framework. Another perspective was discovered by [4] in which ease has been sought with contractual cloud brokers instead of directly dealing with cloud service providers. An interesting dimension to resolving the SLA enforcement was nicely presented by [163] who strongly believes that automated SLA enforcement can easily be dealt with by integrating a feedback control system by filtering any inputs associated service delivery faults such as noise or disturbance, to get the clean measured output, which can work regardless of the cloud environment.

Introducing three components for this purpose such as a rSLA language:, rSLA Service and set of Xlets(a generic REST API for monitoring purposes) which shows a certain level of confidence to achieve the problem for enforcing the SLA [136]. These solutions do focus on the SLA enforcement alone however, their solutions merely touch the security and privacy aspects. The idea rolled over by [202] primarily stresses to Dynamic Bayesian Network (DBN) which gets trained and fed the outcomes to another module Long Short-Term Memory (LSTM) neural network which performs unique analysis on the services with poor QoS to enforce the cloud SLAs. Finally, [217] and other similar ideas suggest solving the SLA enforcement problem with blockchain techniques gaining credible witnesses which are embedded with smart contracts. This approach does appeal however, it also presents some of the challenges, while working and reviewing use cases in the real world.

Fair Exchange Protocol for Cloud Services

Previously studied literature shows many approaches have been introduced towards SLA enforcements within cloud computing, however, considering the SLA enforcement in particular, as a fair exchange problem has rarely been seen [160]. The Fair exchange protocol (FEP) was previously discussed by few researchers in different perspective [24, 33, 161, 196, 197], however in our research context, e.g. FEP's implications with the notion of contract signing in distributed systems, it was well explored and nicely debated by [20, 32, 79, 84, 218]. FEP being a Marvel of e-commerce, enables global business partners to freely trade with each other demonstrating a great sense of trust and confidence. Various operating protocols are opted to manage and supervise electronic trading for distinctive purposes [124, 150, 157].

The entire exchange exclusively spins around ensuring and maintaining a process composing a notion of effectiveness, timeliness, and fairness [10]. Regardless of exchange commodities (e.g. tangible or non-tangible items) fairness is the key element to observe so no one can be treated dishonestly against their contractual commitment and obligations. This refers to sensibly placing a regulatory and monitoring secure mechanism when two or more corresponding business entities cannot purposefully or accidentally cause any kind of deceptiveness by depriving another party of their agreed share or privileges.

Fair exchange protocols are the best fit for the purpose to supervise invisible digital exchanges. Their adaptability to robustly administer electronic exchanges like emails, electronic payments [79] has undoubtedly proven their successful significance. The eventual involvement of a Trusted Third Party (TTP) becomes inevitable, who can play a decisive role to reconcile any disputes or disagreements by supporting their evidence produced by FEP, in the favor of an affected party in the modern distributed and service-oriented cloud computing platform. In the upcoming sections, it will be demonstrated how this research transforms the same idea to gauge cloud-based service delivery and its dispute resolution, placing an online trusted third party (TTP) to resolve any issues by keeping their profile as minimal possible using multiple communication models *(asynchronous or synchronous)*. Previous work where fair exchange protocols were used for only limited avenues such as certified emails, digital signatures, and electronic payments [10, 20].

Previous approaches addressing the problem of SLA management, cover certain service segments by leaving SLA enforcement problems set aside. Their prime focus sticks towards SLA monitoring, detection, which only raises alarms to the stakeholders. The gap is clear when both the service subscriber and service provider would appreciate having full visibility within a service-orientated

architecture. It let them govern the process execution, monitors detection, and collection of key factors such as why, when, and how a service is failed. It further reviews SLA compliance during the service exchange term. It can also detect any level of service distribution such controls can investigate sub-protocols e.g. initiation, setup, exchange, termination, and of course the most significant of all is dispute resolutions to intact and guarantee fairness. There are solutions e.g. deploying a diverse implementation of trusted third parties, auditing deployed web services, inspecting, verification and validation of fair exchange properties are there, however, with the cloud service industry, finding such robust solutions which come as a complete package which ensures both the fairness attributes of service and the security elements, is yet to be introduced.

This chapter highlights various segments of service provisioning within the cloud service spectrum and how the SLA comes into the QoS monitoring frame. The chapter also reviews, how cloud service providers devise SLA monitoring, detection, and anomaly response using their in-house built platforms. Such arrangements do produce SLA violation data thrown to the service subscribers for their information. The SLA violations, if caused by the CSP, then a limited option is made available to compensate the poor QoS, in terms of service credits, most of the time. To earn these service credits, yet a manual process is the only option for CSS, which is not fair to them. They cannot ask for a privilege to have low-level KPIs, so they can justify their RoI at the end of each service term. CSPs most of the time dominate the entire service plan by offering the least privileges to their clients especially how the SLA service terms are measured, monitored, calculated, and eventually billed to the service subscribers. There are chances that the CSP can trick their customers by sharing some arbitrary KPIs using various techniques as they do own these hosting facilities.

The above reviews how these cloud services are monitored and how subscribers are forced to pay their financial commitments to CSPs without having an in-depth QoS reconciliation. Figure 2.3 gives another narrative that how a cloud provisioned service should ideally be monitored at CSS's platform. This is a shared privilege that should principally be offered by the CSP. Bringing the peace of mind for the service recipient, so they can fully understand and justify the service fairness against the money they have paid to the CSP, using some sort of Business to Business *(B2B)* Payment Systems for each term. The chapter contents also elaborate, fair exchange protocol and its capabilities how it can handle SLA enforcement. Various researchers addressed the issue in context of SLA monitoring, detection and reporting such as [46, 71, 121, 136, 149, 151, 163]. Most of the work whirls around SLA monitoring and violation detection mechanisms. Rest [76] proposed a broader

picture by discussing various scenarios while two entities are interacting under fair exchange protocols. Their work motivates our work to consider end-to-end protocol automation minimizing the participant's interaction without compromising a claimant's integrity.

[178] further demonstrates collection of SLA specifications.[49, 214] advocated idea of involving trusted third party(TTP) /auditors who can act as unbiased entities. Finally other approaches were introduced by [1, 201, 209] who used blockchain to manage SLA. Our research takes the narrative to an extra mile by by showcasing how fair exchange protocol can be employed for this purpose. Although our conceptual model presented the protocol's initial phases however upon presenting the next phase of our protocol research we extended participant's scenario from loss averse to malicious by considering *"data security and privacy comes first"*. To consider various security and privacy arrangements such as secure co-processor(trusted modules)[25, 49, 51, 78, 80] , Fully Homomorphic Encryption Schemes *(FHES)* [69, 119, 122, 134, 164] and append-only database [64, 94, 155, 198] were consulted to additional security and privacy layers within our proposed protocol to minimize any potential vulnerabilities.

After embedding various strategic technical security and operational features, we introduce novel architectures which facilitate and intelligently enforce exchange fairness. Its early capabilities towards SLA verification, validation, monitoring, traceable forensics auditing, violation detection, prevention, privacy, and anti-tampering might hold some teething problems and of course, open to a lot of refinements yet the architecture is there to enforce trusted service treaties among multi-parties cloud service exchange. The proposed work also manifests some critical gaps which were noticed in the previous work and facilitates parties to consider a doable trusted solution with full confidence.

This chapter was intended to explore how web services work, their basic architecture, a service life cycle. It also touched what is the significance of a provisioned web service, associated KPI's and SLA management. The fact how CSPs monitor SLAs, how SLA violation occurs, and what are the implications of such service failure or poor service delivery. SLA monitoring, detection, and prevention methodologies were reviewed to highlight the gap. We also analyzed what other researchers proposed in terms of meeting SLA requirements using various techniques and how we believe that SLA enforcement is a fair exchange problem.

Our next chapter examines fair exchange protocols and their various types. It also assesses the protocol's suitability and how it works with or without trusted third parties and trusted authorities alike. It highlights potential constraints, communication overheads, and resilience along with their varying

properties if implemented to enforce SLA and how would it benefit both the service entities such as the CSS and the CSP.

Chapter 3

Fair Exchange

Marvels of cloud computing have been enabling pretty many businesses all over the globe. These features such as on-demand, flexible, instant provisioning, and cost-effectiveness serve smartly integrating within business processes. CC services have made online trading so adaptable for either public or private sectors that these enterprises have motivated others to integrate with these technologies. Digital commerce shows a significant boom to retail e-commerce sales worldwide, touching US$3.53tn[192]. During the pandemic situation, technology based trading has connected unprecedented buyers are being attracted to shop online for all sort of goods and services. Enterprises have also been in the same race when they seeking some outsourcing either aiming between business to business *(b2b)* or business to consumer *(b2c)* services.

When some business functionalities require out-of-the-scope tasks which warrants extra overheads, costs and cannot be delegated internally, service outsourcing simplifies such task handling. Outsourcing is carried out on the basis that interacting entities work within their own trust boundaries and their financial gains are not shared whatsoever e.g. between the service provider or service subscriber. In case a business entity causes any damages then liabilities are claimed in accordance with their service contract, legal, or regulatory framework [31].

While a huge number of online business contracts do take place, service providers and service subscribers do often observe some service delivery hiccups. These hurdles may arise either due to *someone's fault*, *something's fault* or a fault may be inflicted by an attacker who may want to disrupt or fail the business process for a number of reasons. Similarly, while the service delivery is being carried out, either the service provider or the service subscriber may face an awful situation when the other participant is not satisfying the business expectations written in the terms and conditions on their contract. A misbehaving party can try to cut the corner to earn some advantages *(financial,*

technical, or operational) using unfair means.

During such a situation where business cannot pledge 100% trust in other counterparts, it becomes really difficult and challenging to avoid any losses, if a lot of mitigation steps are not taken. A service provider is fully paid, he might try some techniques to trick the buyer. Using such methods they can perhaps save money or resources. This might correlate a situation when a fraudster uses *Salami Techniques* [68] which does not deprive others with huge notable chunks at once but such fraudster gradually steals a very small amount which goes undetected by the victim so at the end of the game the fraudster could pocket a huge sum. This is the point where the economic fairness gets compromised and hit hard the suffering entity.

Cloud computing on one side accommodates such economic growth using its cutting edge technologies and on the flipside raises a lot of constraints that collide with economical ethics, best practices, regulatory requirements. Such corporate level cheating on agreed contracts leaves an irreversible business impact on the victim party. CSS, who is in a need of provisioning some instant, on-demand cloud resources contacts the CSP, who offers their services for this purpose. The cloud service subscriber pays the CSP upfront. CSP on the other end doesn't comply with their commitments at all or they do but the level of the service e.g. QoS is too poor and is not acceptable to the CSS, who demands some kind of compensation. A party believes that they have not been delivered their due electronic items/services whereas another participant can claim they did it because they have not been paid in full. There are also chances that both of them are honest, there also chances one of them isn't and attempts to cheat the other. Prospects are also there when someone with some malicious intent trying to attack them stealthily.

Performing a secure exchange on where betrayal and distrust are ever expected especially using untrusted operating platforms*(nodes, links)*, certainly warrants a robust protocol, which ensures an end to end fairness and also capable of resolving any disputes among buyers and sellers with the least overheads and shortest possible time frame. This research work focuses on examining such solutions and strongly believes that it relates to the problem of Fair Exchange.

3.1 Fair Exchange Protocol *(FEP)*

The term fairness has been used within economic and legal domains for quite some time ago. It narrates some obvious implications towards defining and justifying how the sense of fairness is being maintained while multiple parties

involved with some communal contractual *(verbal/written)* commercial exchange *(goods/services)*. The outcome of such *formalized* trading activity may decide the economic fate either for both the parties been involved, if both are honest or if either of them or both are dishonest. It certainly goes against the party *(victimize)* who acted honestly, however, the party acted another way around to have a bigger slice, using some dishonest maneuvers which deprives the honest participants.

Fair Exchange Protocol *(FEP)* has been well known on security protocol on the horizon perhaps since the commercial trading commenced. The protocol's implementation has also been carried out either formally or informally, through a verbal agreement or as in a written contract to maintain the segregation of services, expectation, performance, length of service, quality of service, and much more. Contracts based upon FEP must have had some sort of elements stating agreements, disagreements, authorization, unauthorized, approved, non-approved, or other similar economic factors to protect each party. We will be discussing and exploring the FEP, in terms, its definitions, scope, types, operating environments, components, participants, architecture, communication channels, interactions, and how it relates to the modern-day service exchange specifically when exchanging digital items (goods/services) online has become a norm.

The work will also be touching the fact that how FEP is associated with the cybersecurity domain and its significance when innovative economic activities are at their boom using secure but *"vulnerable"* hostile operating platforms and so-called trusted but *"untrusted"* where attackers are ever ready and clever than ever so to exploit these channels to gain their malicious intents which could be for financial or state-sponsored reasons. Our work will also demonstrate that business participants while trading interactively could well be loss averse, however, some of the exchange elements could affect their fair exchange. The review would also navigate through to the fact that it's won't take too long for an honest business participant to become dishonest to earn unsolicited huge financial gains by depriving others business partners who put their trust in them.

3.1.1 Defining Fairness

The fair exchange was nicely discussed by an academic who suggests that before setting up a contract, each participant can evaluate the work performance worth for them offered by others if they are considering some exchange. Fairness of exchange is well justified, if both parties *(buyers/sellers)* acknowledge their items then, the exchange must, *by definition, be fair*. Committing some sort of

exchange accountability which proves performance and promises at the end of the contract that, it must be assumed about each party is content with his bargain, or he would not have made it[24]. [161] narrates some assumptions and the possibility of achieving the fairness element if those assumptions are relaxed while two participants are exchanging their secrets. [196] raises the concern how a misbehaving party can halt the exchange by taking advantage.

For a wide range of applications, such as contract signing and e-commerce, where participants require an item from another participant, a problem called *fair exchange* needs to be solved. A fair exchange protocol guarantees that no honest participant will be at a disadvantage should another participant decides to behave maliciously. Intuitively, a protocol is deemed *fair* if no misbehaving participant in the protocol can gain an advantage over other (*honest*) participants. For example, a protocol in which two participants exchange one item for another is fair if it guarantees that either each party receives the item it expects, or neither receives any information about the other's item at the end of the exchange [21]. Such a property is typically termed as *atomicity*. This type of protocol is known as *fair exchange protocols*.

Fair exchange protocols are used for applications that require an exchange of items. Example abound: (i) online payment systems [41, 53], in which a payment is made in exchange for an item of value, (i) contract signing [21], [32], in which the two parties exchange commitments to a contractual text, certified electronic mail [22, 26, 63, 218] among others.

There are several categories of fair exchange protocols, which are:

- Gradual exchange protocols [32, 43]: This type of protocols works by having the parties release their items in small installments (i.e., gradually), to ensure that the amount of knowledge on both sides is approximately the same at all times during the protocol execution. This means that, typically, no party has an advantage over the other. Of course, because the release is gradual, such protocols suffer from a large number of communication steps. Gradual exchange protocols are also not suitable is situations, where the items to be exchanged, have a "threshold" value (i.e., the item may be valuable or not).

- The second class of fair exchange protocols works on the premise of the existence of a trusted third party in the system [63, 218], which also addresses the threshold problem. The trusted third party typically monitors the communication between the protocol participants, ensuring that no participant receives the item it wants before releasing its item, ensuring fairness. The main disadvantage of the trusted third-party

solution is that, as it monitors all communication steps, it may become the communication bottleneck if it has to be involved in all instances of the protocol to guarantee fairness. Variations of this approach include fair exchange protocols with a semi-trusted third party [79].

- To address the communication bottleneck of trusted third parties, a class of protocols have been proposed for what is known as *optimistic fair exchange* [21, 28]. While these protocols still require a trusted third party, the third party is only needed when there exist deviations from the normal protocol execution such as when messages are delayed or one of the parties misbehaves. In short, the trusted third party is only involved when problems exist in the system.

A typical specification for a fair exchange problem consists of the following properties [10]:

- Effectiveness: If the protocol is executed properly with the parties A and B being honest participants, then both parties will have each other's items at the end of the protocol execution.
- Timeliness: The protocol will complete infinite time.
- Fairness: There are two flavors of fairness: (i) strong fairness states that, at the end of the protocol, either both parties receive the expected item from the other party or no party gains the expected item, or (ii) weak fairness states that at the end of the protocol, either strong fairness is satisfied or the honest party that do not receive the expected item does not lose out, i.e., not receive the expected item.
- Non-repudiation: After a protocol run, each participant P will be able to prove the origin of the item it has received and prove that P's protocol counterparty has received P's authenticator.

FEP classifies fairness as *strong* when both the exchange participants receive each other's item or none do. Such scenarios establish the level of confidence among the exchange participants because of the minimal risks [20, 81]

FEP is supposed to be *weak* when either of the exchange participants does not receive the other's item but can prove(to authority) that either the other participant has received their item or they can receive the same without any further assistance from the victimized participant, it would be weak fairness. This scenario facilitates dispute resolutions through enforcement mechanisms.

Weak fairness is especially important as it allows an honest participant to initiate a post-exchange dispute resolution in the case it has not received the

expected item. As such, protocols that guarantee weak fairness will make use of a trusted third party (TTP) to help with dispute resolution[20, 81].

Several protocols that solve the fair exchange problem have been proposed in the literature; for the reason of completeness, we provide one from [76], that includes a post-exchange dispute resolution phase. The protocol is found in Table 5.1. More details about the protocol, including proofs of correctness, can be found in [76].

The fair exchange protocol in Table 5.1 consists of three interdependent subprotocols: (i) a normal *exchange* protocol (E), (ii) an *abort* protocol (A) and (iii) a *resolve* protocol (R). The CSP and the CSS start the exchange by executing the exchange (E) subprotocol. If they are both honest (and as the system is synchronous), then they will exchange the items after the end of the exchange protocol. If, for example, one participant is delayed in replying to a message (i.e., violates timeliness assumptions), then the other participant may execute the abort protocol (A). On the other hand, if the item received by one of the parties is not according to the agreed description, then that participant will execute the resolve (R) protocol. Note that an occasional violation of synchrony assumption leads to either the execution of the resolve or abort sub-protocol. In the latter outcome, as in transactions, enforcement can still be done through re-execution or dispute resolution through TTP. Thus, the fair-exchange based approach effectively facilitates automated enforcement of SLA violations.

Table 3.1 gives a bird's eye view covering some fair exchange protocols along with their obliged service scope, properties, dispute handling capabilities, trusted third party involvement, trusted authorities/ certificate authorities, and their deployment. This brief analysis presents a good picture, entailing FEP trends and what potential compromises a deployment would face achieving most features such as fairness, timeliness, effectiveness, non-repudiation, and service overheads. To compare these protocols with ours, we have also extended the study to understand that how previously introduced FEPs can support forensics investigation in case low-level message exchange logs are warranted to resolve a dispute between the CSS and CSP, in the presence of TTP and their coordinating TAs.

Chapter 4, examines system models and fault models representing the cloud computing environment. It further explains that how processes, links, and business entities can act while they are employing digital services against some monetary commitments. Their behavioral analysis can lead them towards either loss aversion or participants could act maliciously. Fault models do help to highlight potential threat points, which we have included in our proposed architectures.

No	Proposed Schemes	Protocol's Features														DeploymentType
		Exchange Items	F	T	N-R	S	P	FDH	CF	SLA_F	S_e-F	E2E S-Os	FL	PR	TTP	TA	
1	Rabin1981howte	digital item	Y	N	Y	Y	Y	N	N	N	P	Y	S	N	N	N	N/A
2	burk1990value	digital item	Y	N	N	N	N	N	Y	Y	Y	Y	S	N	Y	Y	In-line
3	Jakobsson95	digital payments	N	N	N	Y	Y	Y	Y	Y	Y	N	N/A	Y	N	Y	N/A
4	asokan1997optimistic	digital payments	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	S	Y	Y	Y	Y
5	zhou1997evidence	digital items	N	Y	Y	Y	N	Y	Y	Y	Y	Y	W	N	Y	Y	In-line
6	franklin1997	digital payments	Y	N	Y	Y	Y	N	N	N	N	Y	W	N	Y	Y	On-line
7	pfitzmann1998optimal	exchanging signatures	Y	Y	Y	Y	Y	N	N	N	Y	Y	W	Y	Y	N	Off-line
8	asokan1998asynchronous	digital signature	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	S	Y	Y	Y	Off-line
9	Pagnia1999OnTI	digital item	Y	N	N	N	N	Y	Y	Y	N	Y	S	Y	N	N	N/A
10	Gartner1999	digital item	Y	Y	N	Y	N	N	Y	Y	Y	Y	S	N	Y	N	In-line
11	ShmatikovM00	online payments	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	S	N	Y	N	on-line
12	ray2002fair	digital item	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	S	N	Y	Y	In-line
13	nenadic2004fair	email	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	S	Y	Y	N	Off-line
14	nenadic2005rsa	digital items	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	S	Y	Y	Y	Off-line
15	Ray2005	digital payments/ items	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	S	Y	Y	Y	Off-line
16	0002MM06	digital payments	Y	Y	N	Y	N	N	Y	Y	N	Y	S	N	Y	Y	On-line
17	ZhangSMA06	digital item	Y	N	Y	N	Y	N	N	N	N	Y	S	N	Y	N	On-line
18	devane2007secure	digital payments/ items	Y	N	Y	Y	Y	N	Y	Y	N	Y	S	N	Y	N	On-line
19	Guttman2012	digital payments/ items	Y	Y	Y	Y	Y	N	N	N	Y	Y	S	N	N	N	N/A
20	dashti2012efficiency	digital items	Y	Y	Y	Y	Y	N	N	N	Y	Y	W	Y	Y	Y	On-line
21	kupccu2013distributing	digital payments/ items	Y	Y	Y	Y	N	Y	N	N	N	Y	W	N	Y	Y	On-line
22	Huang2015	digital signature	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	W	Y	Y	Y	On-line/ Off-line
23	Wang2016	digital payments/ items	Y	Y	N	Y	N	Y	Y	Y	Y	N	W	Y	Y	Y	Off-line
24	seo2018accumulable	digital items	Y	Y	N	Y	N	Y	Y	Y	Y	N	W	Y	Y	Y	On-line
25	wang2019optimistic	digital payments/ items	Y	Y	N	Y	N	Y	Y	Y	Y	N	W	Y	Y	Y	Off-line

Table 3.1: Fair Exchange Protocols Comparison

F=Fairness, T=Timeliness, NR=Non-Repudiation, S=Security, P=Privacy, FDH=Facilitates Dispute Handling, CF=Cloud-Friendliness, SLA_F=SLA-Friendliness, S_e-F=Supports e-Forensics, E2E SCO=End2End Service/Communication Overheads

Chapter 4

System and Fault Models

4.1 System and Fault Models

System and fault modeling is a technique which lets one evaluate, various kind of system failures such as defects(imperfections in the infrastructure), faults*(imperfection of some functionalities)*, errors*(an incorrect system behaviour)* and failures*(deviation of a specified behaviour)*. Critically reviewing a system model against its corresponding fault models, surely opens various corridors to fix those problems, and also it methodically assists to have detailed assessments of those fault points and their most appropriate mitigation and improvement approaches. In a testing scenario, a system model may include participating entities in their good operating state with some fundamental implementation configurations. Fault model, on the flip side, discovers potential fault points, from the beginning of the process till the endpoint, showing potential misbehaviours of a single node or a distributed computing cluster. That's system-level fault identification and fault points are analyzed through the chain of events. [72, 159]. This chapter presents some thoughts by discussing our prescribed system model, fault model scenarios covering both our proposed architectures for enforcing cloud SLAs, hence, it also constructs an overall security posture to correlates enforcement elements. This research further extends by evaluating every main component of said architectures by furnishing mitigations schemes for discovered problems and constraints. Following we present these models, we assume in the published paper.

4.1.1 *When Participants are Loss Averse*

The following discussion covers our first proposed architecture, which addresses a use case when interacting participants are classified as loss averse as explained in the Chapter 5, who interact through a synchronous communication model.

System Model

Within distributed computing, a system model consists of all the participating nodes, processors, processes, and their associated components interactively communicating with each other in various operating environments. Their coordinated communication facilitates performing a task such as (a client's) service request initiated by one of the expected processors who is a member of a known network coming from either a trusted or untrusted network. The system model also illustrates some algorithms or protocols under which at least two or more interactive participants agree on their interaction cycle that how their communication will set up, initiated, task performance, and expected behaviours and terminations, once the tasks are completed. Issues related to fault tolerance and dispute resolution are also put in place within the service reference architecture. It facilitates review of the message exchange or expected communication pattern, timing model when participating nodes are interacting with each other regardless of their service distribution span.

The system model also reviews every component functionality as expected and their obligatory behaviour when working in an ideal environment. Evaluating how long a process takes a time for execution and complete the task depends if the system is set for running those tasks on either an asynchronous and synchronous communication models as both have their operating implications [52, 116]. Further elaborating the fact, our chosen communication model is synchronous therefore upcoming interactions will be reviewed under the same assumptions. Synchronous systems do depend on real-time so to communicate with correct processors[54].

We assume a system with three parties: *(i)* the cloud service subscriber (CSS), *(ii)* the cloud service provider (CSP), and *(iii)* the end-users.

CSS: The cloud service subscriber is an entity (e.g., individual, company, government agency) that requires computational resources such as processing power and/or storage. The number of resources required by the CSS will be determined by the application workload that the CSS will be expecting for its business requirements. In return for these computational resources (and other associated services, such as disaster recovery), the CSS will provide an item such as digital payment in return. The amount of computational resources required by a given CSS is dictated by the application or business requirement of the CSS. For a given CSS, we denote its application requirements by A, which contains a set of (business or application) parameters such as availability and response times among others, together with desired values. We assume that A can be translated into the corresponding resource requirements R that the application will require from a CSP, i.e., the CSP will be able to map A

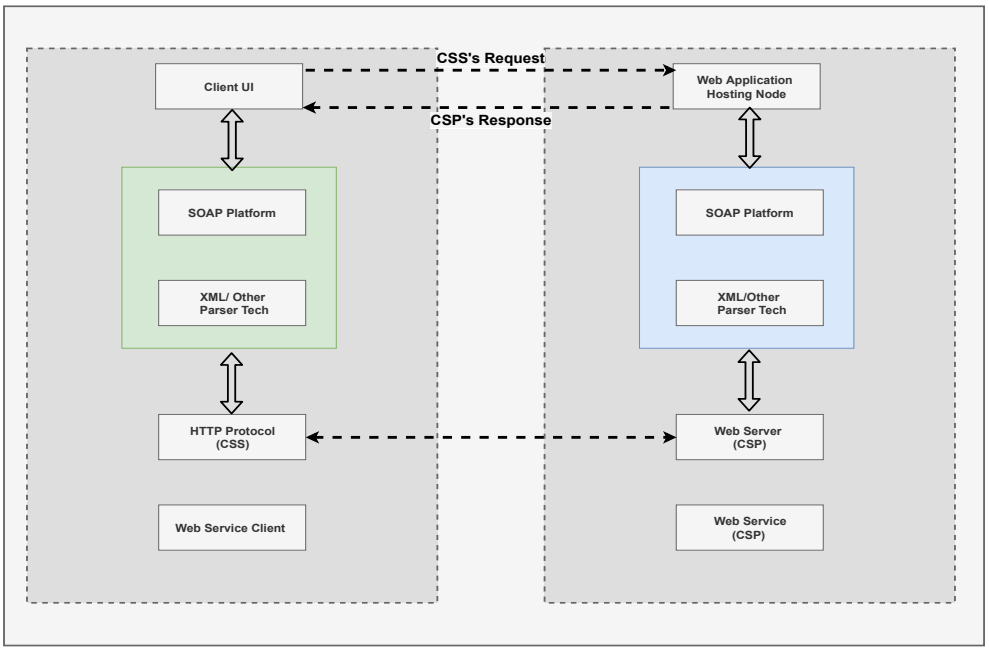


Figure 4.1: A basic Web Service Operating state [35]

onto a relevant R. We assume that the CSS has a function E that takes a set of metrics (or sensors) S from the CSP to determine whether $E(S)$ satisfies A. We also denote the item given by the CSS by I_{CSS}. We assume that the item I_{CSS} satisfies some specification $\Sigma_{I_{CSS}}$ and that there exists a verification procedure $V_{I_{CSS}}$ that can attest whether I_{CSS} satisfies $\Sigma_{I_{CSS}}$ or not.

CSP: The CSP, at the lowest level, provides computational resources to one or more CSSs. The CSP uses a variety of supporting techniques such as VM migration [36, 183, 213], scheduling [113] and elastic resource provisioning [48] to meet the resource requirements of every CSS requesting its services. The exact CSP deployment is outside the scope of this paper. (To hint, given a CSS and its application requirement A, the CSP needs to decide, based on the resources it has available and its supporting techniques, whether it can allocate R resources to A, e.g., see [75]). We assume that the CSP keeps track of a set S of *signals* (or sensors) for the CSS which, together, captures the performance of the CSP towards meeting A. Due to competing demands from various CSS's, a CSP may not always guarantee that their respective requirements will be met at all times. To capture this aspect, the CSP provides a corresponding SLA to the CSS, which we denote by SLA, that stipulates the performance guarantees that the CSP provides to CSS. The SLA will proclaim clauses for each application parameter of A. We refer to each clause as a (parameter) *predicate*. When any such predicate is violated (after evaluation by E), the CSS may escalate remedial action to the CSP, which is captured in SLA. We assume that, periodically, the CSP pushes the value of S to the CSS

for evaluation.

Message Security: Arrangements & Schemes While discussing the system model it will be nice to review and explore some of the supporting technologies protecting (e.g. data security and privacy) message exchange channels and act as mandatory security implementation. This becomes even more critical for parties to review how their service providers do care about security threats, attacks, and other aspects to ensure secure service delivery to their client's platforms.

Cryptographic Primitives Cryptography is one of the prime elements when ensuring communication security to/from untrusted channels. Surely, without implementing cryptographic schemes data security and privacy can never be guaranteed. Furthermore, these security methodologies do serve security elements such as confidentiality, integrity, authentication, and non-repudiation. The same has been adopted in our work so while assuming message exchange or other party communications, our protocols do rely on cryptographic primitives therefore, the thesis does require to shed some light on various cryptographic schemes and how they encrypt and decrypt messages. A message is encrypted when a rendering process is implemented on some intended data which makes that data unusable (e.g.no one else can read except the intended users/ recipients). These schemes prevent unauthorized actors to read that information being sent/ receive *(data at rest, data in transit).* An attacker can even access the encrypted data, they cannot decrypt it until they have access to appropriate cryptographic keys [200]. This message transformation from plaintext to ciphertext where the order of message bits are scrambled to make it unreadable for unintended and unauthorized users until the attacker can attempt by reverse engineering those encryption rules and cryptosystems. There are few techniques, we will be reviewing to understand the protocol communication and security assumption when advancing the participant's inter-communications arrangements. Table 4.1 presents various schemes along with their attributes and objectives [187, 200]

Table 4.1: Cryptographic Schemes used in message exchange

Cryptographic Schemes			
Encryption Scheme	**Objective**	**Input**	**Output**
Symmetric cryptography	Confidentiality	Plaintext(any lenght)	Ciphertext(same lenght)
Asymmetric	Authentication	Plaintext(any lenght)	Fixed lenght signature
Asymmetric	Key agreement	Counterparty information	A session Key
Hash	Fingerprint	Plaintext(any length)	Fixed length message dependent fingerprint
pseudo-random number generators(PRNG)	Randomsness	Various	Hard to predict bits

Symmetric Encryption The symmetric key encryption method also known as secret key encryption, uses a single secret key for both the sender and the recipient during the intended message exchange so to perform both encryption and decryption, using the same key. The message sending entity encrypts the message using this key and the recipient on the other end decrypts the received encrypted message uses the same key to decrypt it using some back-end mathematical hashing functionality. The security concern for implementing this method is not very reliable as if the secret key which needs to be shared with the receiving party, can be intercepted maliciously by some attacker, the security and the integrity of the message may compromise[200]

There are some algorithms known to the Symmetric family as Tiny Encryption Algorithm *TEA*, Data Encryption Standard *DES*, International Data Encryption Algorithm *IDEA*, RC4 and Advance Encryption Standard *AES*.

Asymmetric Cryptographic Algorithms This methodology which is also known as *public key encryption* uses a different scheme for encryption and decryption such that a message can be encrypted by user A by using user B's public key (might be known to others e.g. unauthorized users) and the message is dispatched to user B. In terms of decrypting the message, user B has to use their private key(only known to user B, not to anyone else) otherwise, a message if may be intercepted by a middle attacker, they cannot be able to read it as they don't hold a private key from user B. The basis on *one-way functionality* uses an approved mathematical operation which calculates such values with a combination of inputs, however, it is impossible to regenerate

the input values again. *Diffie-Hellman* and *RSA* are well known to this family.

After reviewing various studies such as [145, 154, 175], and other researchers such as [23, 27, 67, 125, 150, 205] who have mainly worked used this algorithm, also [182] survey shows that RSA is the most appropriate algorithm to work within the cloud computing environments, therefore, our chosen algorithm for the context of our research will be based on RSA algorithm.

Hybrid cryptographic Methods Usage of Hybrid Cryptographic algorithms are another emerging idea specifically for e-commerce, such introducing a combination of a symmetric key algorithm of AES and the asymmetric key algorithm [107] and another variance [117] introduces hybrid cryptographic combo with two different symmetric algorithm using simple integer variables and extended linear block cipher to ensure integrity, confidentiality, and authenticity while the participants are interacting online.

Digital Signatures Digital Signatures: Digital signature is a compiled hash value that is signed by the message senders using their private key, which ensures the message's integrity. The industry has been using strong digital signatures on various digital products. When dealing with sensitive items or online transactions, signing them digitally adds surety about its protection in terms of message integrity and non-repudiation and authentication will be maintained between senders and receivers. The significance of digital signatures certifies data exchange is trustworthy as the message exchange gets back and forth on untrusted channels. A digitally signed piece of work is intuitive puts incredible challenges to the attackers although in terms of attempting some arbitrary modifications to the data, however, scenarios like double spendings [52, 93].

Non-cryptographic Methods Other significant security measures on top of cryptographic algorithms are traffic padding, routing control, passwords, smart cards, protected channels, and then other obvious security arrangements such as firewall are also implemented to ensure channel security [120].

Nonce is a one-time usage integer value that is included in the message sequence to demonstrate the message is not a repetitive message. Nonce puts another security layer for an authentic message to be used again by the attacker as *replay attack*.

Digest Functions are also known as secure hash functions and some time they are denoted as *H(M)*. The fact while reviewing in a message exchange both H(M)≠H(M') needs to be highlighted and should be treated as modified message, while the participants were interacting [174].

RSA

No doubt that in recent years RSA *(Rivest, Shamir, and Adleman)* [37] has been proved itself as the industry's de-facto cryptographic algorithm when implementing public-key(asymmetric) cryptography. It does not only support the protocol's authentication requirements but also performs encryption core functionalities. Following explain the basic encryption and decryption process in following steps where performs key generation, encrypts the intended data and finally decrypts it: [52, 154]

1. A Message Sender*(A)* intends to send a message to a recipient *(B)* so he generates a key pair K_{pubA} and K_{privA}. The public key K_{pubA} is published in a desirable secure location.

2. Now *(A)* computes a digest of message M as M, H(M) using approved some secure hash function and performs the encryption using his private key K_{privA} so to get message signature (S) as S=H(M)K_{privA}

3. Sender *(A)* dispatch the intended encrypted message as [M]K=M, S to the message recipient *(B)* through untrusted communication channels.

4. *(B)* performs the message decryption function on S through K_{pubA} and computes the message digest of M, H(M), if both matches, the signature of the message can be classed as valid.

Despite being a well-reputed encryption algorithm for quite some time, RSA still has got few security concerns such there are chances that the entire data can entirely be decrypted if an attacker can get hold of the private key. This threat can only be mitigated with algorithms supporting *forward secrecy* feature such as Diffe-Hellman Encryption*(DHE)* however, DHE got its issue such as its speed. Another best option would be implementing Ephemeral Elliptic Curve DHE *(ECDHE-RSA)* is on asymmetric encryption catalog which is robust and fast [123, 173].

Digital Certificates & Certificate Authorities *(CAs)*

Digital certificates offer an assurance that they are the legit parties to whom the communication is being carried out. Digital certificates hold public keys and some identification followed best practices or some global standards of a remote user, which by repelling attackers and unauthorized users for being impersonating. A certificate may include a version of X.509 *(maintained by International Telecommunications Union)* certificate's serial number, signature algorithm identifier issuer name, validity period, subject's name, and subject's public key which proves a good acceptable identification. Certificates are issued by certificate authorities *(CA)*. Such organizations offer digital notary services

to interested parties so they can get digital certificates only from approved CAs [93].

Fault Model

Fault models are developed to forestall an operating situation where either one or more system participating nodes, processors, processes, links, or any of the contributing service components may fail, due to multiple internal or external factors. This outcome due to such faults can end up calculating and presenting inaccurate processing times, (system, process, verification)values, falsified specifications. Such faults may also interfere with system security configurations and infringement of user access, authorization, or any other aspect which demonstrate as some sort of arbitrary action from one of the serving nodes [108]. Further analysis discussing such failures [92] well describes by detailing about e.g. omission (process/communication channel)failures, arbitrary *byzantine* failures, and timing failures on top of those which are classified as benign, intermittent, and transient faults occur during message exchange.

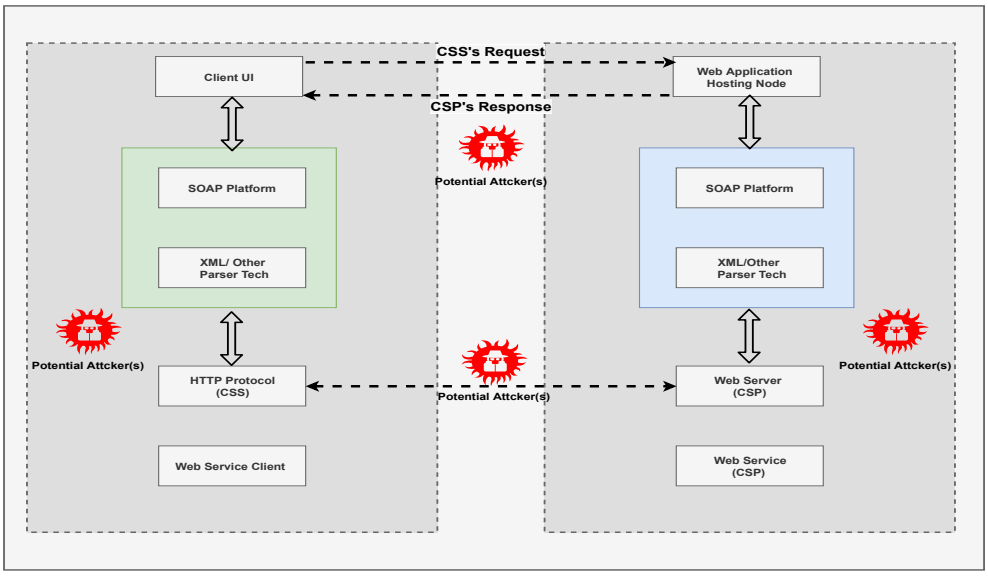


Figure 4.2: Potential attackers could cause failure to a web service operation [35, 73, 140]

Focusing on the core web technologies where web services are interacted such as published, accesses, or even attacked, security measures are put in place to ensures:

a) *Message Security*
b) *Infrastructure Security*
c) *Correct Response Generation*
d) *Authentic Recipient Only*
e) *Interaction Complies the SLA*
f) *Repel Attackers (internal/external)*

A SLA violation alert can be triggered if one of the service elements on CSP's provisioned services based upon either IaaS, PaaS, or SaaS environments, as mentioned in Table 2.1 are compromised

The fault model we assume in our published paper refers to those faults which can cause a CSP to be unable to fulfill its SLA concerning the CSS, i.e., we focus on faults that cause $E(S)$ to violate A. This could be due to a failure of the CSP's supporting techniques *(i.e., design faults)* or its misconfigured resource provisioning *(i.e., component faults)* or even it could be due to a malicious attack *(i.e., external abuse)* such as a distributed denial of service *(DDoS)* attacks. Of interest here are the manifestations of such faults, i.e., SLA violations.

We also assume the CSS and CSP to be *loss averse*. A participant *(CSS or CSP)* is loss averse if (i) he does not attempt to subvert the execution of the protocol, (ii) if the participant is a CSS, then the participant always correctly evaluate $E(S)$, i.e., the CSS does not falsely accuse the CSP of violating the SLA and (iii) if the participant is a CSP, then the CSP always agrees to the remedial action specified in SLA when the SLA has been violated. The reason for a loss-averse participant is that (i) as a CSP, a bad reputation may negatively affect its business plan, and (ii) as a CSS, wrongful accusations may lead to reduced availability due to the time for violation resolution.

Nodes and Network: We assume that the node on which the CSS executes is reliable, i.e., the CSS always can monitor the performance of the CSP. We also assume that the clocks of the CSS node are synchronized to real-time within a known bound. WLOG, we assume the bound to be 0 for simplicity. Further, since we focus on faults and attacks that affect the CSP, we assume that the network is reliable, i.e., the delay between the CSP and CSS is bounded and the delay between CSP and end-users is bounded too. In short, we assume the system to be *synchronous*.

4.1.2 *When Participants are Malicious*

In the above-mentioned scenarios, the scope of our system and fault model was only limited to three participants *e.g. CSS, CSP, and the end user* however, as we extend our protocol operating assumptions hence the number of participants are also increased in this interaction. Now we other participants as some of them are well connected as they do interact directly with each other whereas some of them do act as either on behalf of the front-line actors such as CSP or CSS. Those extended participants are trusted third parties (TTP), trusted authorities (TAs). These trusted authorities act as guarantors on behalf of those entities who have outsourced them for this purpose. TA_{CSS} and TA_{CSP} are obliged and given a specific mandate to interact with TTP when they do

esquire or ask to submit their items, pledged by either CSS or CSP. Therefore, the dimension and the scope of our system model and fault model will take a slight shift including these participants. Most of the above-mentioned operating constraints may be the same, however, due to some obvious changes in our architecture and other interacting scenarios, the scope of the fault model will ultimately be extended.

Threat Landscape The SLA violation occurs when one of the deployed web services either gets compromised by a malicious actor. It could be either someone *internal/external* attacker or something*(e.g. a modified process, breached security)* which can jeopardize either a service component or the entire service can be down. In our research scope faults can be occurred within CSP's trust boundary, CSS's trust boundary, or on one of the communication channel, they are interacting with.

Cloud Attack Vectors Cloud migration has seriously accelerated through the globe, regardless of business size, nature, or service scope. During the recent pandemic era, decision-makers would make cloud-based technologies their first and far most choice, which facilitate them a least maintained service platform with the excellence of resilience, security, and compliance. Entire global business has become fully dependent on cloud technologies where the supply chain management relentlessly collaborates international trading, education, shipping, health, financial technologies(fintech), food processing, Wholesale and Retail Trade, and of course manufacturing sector when access to the workplaces have been restricted. To run these sectors hosting technologies been constantly moving towards cloud data centers, where these cloud service providers revenue growth would dramatically increase touching $304.9 billion figure in 2021, with software as a service (SaaS) holds the top position as the largest service element [191].

This depicts the global business reliance on cloud technologies, which unfortunately also attracts bad actors, regardless if they reside within the trusted boundaries or outside the trusted boundaries. Their aim to ceased targetted cloud services in one way or the other by employing various attacking tools and paths. Confidentiality, integrity and availability attack vectors are designed and target [97] critical data*(residual or wired)*, edge computing infrastructure(physical or virtual) or even endpoints. This poses constant security risks to those who are either cloud security service providers or service subscribers. Diversified threat perception can cripple security arrangements and bring a business down to its knees if one of its critical services is compromised. Attack vectors demonstrate their capabilities to an extent that could cause

cloud service outage in the worst scenario. Such ferocious attacks could also cause human lives when health [156] or aviation services [50] are attacked by unknown attackers. SLA compliance and other service monitoring implications would act as double damage when the service outages[88, 170] are waged through these attacks and the service subscriber cannot even prove their claims in these circumstances.

There is a variety of attack vectors which malicious attackers can launch aiming web attacks, application attacks, infrastructure attacks [9, 38, 210]. We can further distinguish such attack vector by service models such as attack targeting SaaS cloud [141], PaaS cloud [215] and attacks impacting IaaS platforms[147, 166, 206].

For the cloud service deployment, the scope of security should consider how data service classification and accountability are arranged. Depending upon the service deployment further threat avenues would also be reviewed before signing the SLA. This would include but is not limited to client & end-point security, service access controls and hosting platform security, marking and agreeing with the responsibility zones. Some of those attacks are being highlighted here :-

a) *Denial of Service (DoS) Attacks*
b) *Distributed Denial of Service (DDoS) Attacks*
c) *Cloud Malware-Injection Attacks*
d) *Cloud Side Channel Attacks*
e) *Authentication Attacks*
f) *Man-in-the-Middle Attacks*
g) *Trust & Reputation System Attack*
h) *Cloud Service Containers (DDoS) Attack*
i) *Metadata Spoofing Attacks (WSDL service modification)*
j) *Server-side Request Forgery*
k) *Credential Stuffing Attack*
l) *Fake Cloud Services*

Chapter 4 discusses system models and fault models and talks about varying service elements whether they belong to technical, operational, or security domains. The chapter also highlights various mitigation options to those risk factors when multi-party service exchange is being performed. We understood how loss aversion would interfere with these business participants and the dispute handling issues. The chapter also discusses those scenarios when either the participant or even both could act maliciously with diverse cloud-based attacks. Our next chapter explains how our proposed conceptual model works, how it initiates message exchange in the active presence of trusted third parties.

The chapter reviews every single message exchange right from the beginning till the end of the service term. It explains how automated SLA enforcement can be achieved.

Chapter 5

SLA Enforcement with Loss Averse Participants

Loss aversion is an economic behavioural condition which indicates when a business entity prefers a business state of no profit no loss, rather then earning such profit which could end up paying them a greater cost in terms of fines, penalties, reputation through media trial, criminal investigation or even court cases. Their sensitivity about avoiding losses is greater value over earning some predicted business profit. This economic move temporarily convince an entity to chose potential negative outcomes over positive outcomes through business gain [110, 126, 204]. On the same note, our first intended scenario is the one, where are both our prime business partners are strong believers of the loss averse theory. Either the cloud service provider who is selling their digital products for cloud environments or the cloud service subscriber who is the potential customer for these digital items would utmost stick to the fact, where they do not reach into a cheating state. Furthermore, the cloud service provider would surely act with full honesty when provisioning cloud services to the cloud service subscriber and would not cheat the CSS such as interfering with their data security, privacy or by making any move which qualifies as a act of misbehavior. Similarly, CSS on the other end would not demonstrate any act such not paying the service provider or not interfering any of the product or service security arrangements, which can be correlated as malicious act.

This research work first highlights the problem and indicates those gaps within the existing cloud service delivery. Related system and fault models, were also discussed in the previous chapter 4 in the first half stating those scenarios how system models work and what happens when potential faults are injected. This leads the research work to propose an architecture that resolves the problem. We suggested that this problem belongs to the fair exchange where SLA enforcement has been a big issue. Previously, the work was mainly

focused on SLA violations and their monitoring aspects. We understand that an intelligent solution is the strategic business need when cloud SLA enforcement is not only monitored, detected but should also be enforced automatically. To the best of our knowledge, the fair exchange protocol has not been considered for this purpose as we did in [160]. This research work presented not only theoretical implications of this suggested architecture but also enclosed with a micro-level implementation using concurrent cloud technologies where the architecture was tested and results were obtained too. Furthermore, we also present details where both the logical and the physical implementation shows how the architecture works and enforces the SLA while using an inline trusted third party *(TTP)* who governs the entire service exchange as sub-contracted by both the CSP and the CSS, leaving least chances of any misbehaviour and loss of fairness from an end to end service transition within the contractual service delivery term.

5.1 Proposed Methodology

Cloud computing (CC) is becoming increasingly important as the economic and technological benefits of such a computing paradigm are evident, with a rising number of cloud service providers (CSPs) offering diverse cloud-enabled services. The growing number of cloud service subscribers (CSSs), due to fringe benefits e.g. disaster recovery or flexible resource availability, seek service guarantees from CSPs regarding the quality of service (QoS) they expect to improve their business case.

An SLA is a primary source to quantify QoS. It originally serves as both a service blueprint and as a warranty for the CC. Such SLAs contain many different clauses that address issues such as (i) ownership of data *(e.g., Access to the data - data retrievable from CSP in readable format)*, (ii) the specific parameters and minimum service (e.g., Availability *(e.g. 99.99% (peak), 99.9% for (off-peak) times or Performance (e.g. maximum response times))* (iii) system architecture and security levels or standards (e.g., Security / privacy of the data *(e.g. encrypting all stored and transmitted data or disaster recovery expectations (e.g. worse case recovery commitment)*, (iv) the costs associated with each level of service, and finally (v) each element of the service, as well as remedies for failure to meet those requirements *(e.g., dispute mediation process (e.g. escalation process, consequences))*. Thus, the SLA captures the guarantees the CSP provides to the CSS, who in turn lease the services against monetary commitments.

However, the system architecture that supports various CC services is

such that it can be (i) subjected to failures, i.e., nodes can crash or be taken offline [12] or (ii) it could fail due to targeted cyber attacks [85, 133]. In these circumstances, the economic impact is often significant to businesses and usually, the CSP breaches its SLA. The clause in the SLA that addresses SLA violation is then activated. Eventually, the CSS, being the penalized party, may seek some form of business reimbursements, as mentioned in the SLA. A manual dispute mediation process could take longer than anticipated to resolve the dispute among participants. Due to restricted access on CSP's computing estate, a CSS is surprisingly held liable for a failed cloud service, when they cannot even produce a piece of evidence. The CSS ultimately needs to adhere to some well-defined procedures: e.g. robustly tracking all relevant data, ensuring its admissibility and integrity *(digital forensics soundness)* is fully intact[2].

To address the problem that the burden of proof rests with the CSP (with all the steps that need to be adhered to), a move towards automating the dispute resolution process will ease the CSP. This entity ensures that either of the participants are not unfairly treated. In this published paper, I propose a state-of-the-art solution based on the concept of *fair exchange* [21, 76].

In the presented paper, my main contribution was to propose a modular architecture that uses the fair exchange problem to solve the SLA enforcement problem. The proposed architecture is depicted in Figure 5.1.

5.1.1 Architecture Objective

In this section, we briefly explain the problem that we address in this paper and subsequently enumerate the properties that a solution to the problem needs to satisfy.

The cloud service fabric is mainly built around a service-oriented architecture defined by the CSP. Typically, the CSS pays the CSP for some services at *stated* times, i.e., periodically. An SLA is then drafted that contains clauses that both CSP and CSS need to satisfy at all times, else remedial actions may be taken against the violating party. The SLA is the only legal instrument that holds the entire service plan, service scope, scalability, outages, and incident response mechanism between the CSP and the CSS. A service variation e.g., under-provisioned services *(in terms of pre-agreed number of processors, memory, storage, bandwidth allocation & service uptime guarantees)* could negatively impact a CSS if users accessing the CSS's application are delivered unfair or degraded services by the CSP. We assume that the CSP keeps track of a set of *sensors* that monitors the various SLA clauses. In such cases, the CSS may wish for some form of compensation whenever the SLA is violated.

To summarise, the scenario is as follows: The timeline is split into "windows" or rounds of equal size (i.e., to capture the notion of the period). Then, close to the end of round c, the CSP sends sensor readings gathered over round c to the CSS. The sensor readings capture the various attributes related to the SLA. If the SLA is satisfied, then the CSS pre-pays for cycle $c+1$. On the other hand, if there is an SLA violation in round c, then the CSS can claim compensation from the CSP (e.g., by duly reducing its pre-payment for cycle $c+1$ services). Since a CSP is assumed to be loss-averse, it will compensate a CSS, so long as it sees the evidence for its SLA violations essential for the CSS to be compensated.

Thus, the burden of proof, i.e., proving SLA violation, rests with the CSS [30]. Such proofs may be very challenging to obtain or show, since the CSS may not have all the required information. Even if a piece of evidence can be found, the process is a manual one, resulting in a long and tedious process that is detrimental to the CSS. Thus, it would be beneficial to the CSS if this process of SLA enforcement is automated. Specifically, automation would benefit all parties, with various evidence captured and presented as necessary. So, SLA enforcement is formulated as a fair exchange in which CSP obtains its pre-payment from a loss-averse CSS for cycle $c+1$ services if and only if it has satisfied the agreed SLA for cycle c services. The fair exchange also has a built-in dispute resolution mechanism: if SLA violations had indeed occurred during c, then both the CSS and CSP receive a token of resolution wherein SLA violations are verifiably acknowledged.

As such, the problem we solve in the paper, which we term as *SLA Enforcement*, is defined as follows:

Definition 1 (SLA Enforcement)**.** *Given a SLA $\mathcal{S}$ that captures the following:*

- *A set of sensors at the CSP denoted by I_{CSP},*
- *An item to "pay" the CSP denoted by I_{CSS},*
- *A specification Σ_{CSP} that describes the range of allowable values of every sensor in I_{CSP},*
- *A specification Σ_{CSS} that describes the range of allowable values for I_{CSS},*
- *A verification procedure $V_{I_{CSP}}$ that verifies whether the value of every sensor matches that specified by Σ_{CSP},*
- *A verification procedure $V_{I_{CSS}}$ that verifies whether I_{CSS} matches Σ_{CSS},*
- *A time period T_E which captures the period over which the values of sensors are valid,*

then a protocol $\mathcal{A}$ solves the SLA enforcement *problem iff $\mathcal{A}$ satisfies the following:*

- *Fairness: When $\mathcal{A}$ terminates, then either (i) CSP receives I_{CSS} and $V_{CSS}(I_{CSS})$ and CSS receives I_{CSP} and $V_{CSP}(I_{CSP})$ or (ii) if $\neg V_{CSP}(I_{CSP})$ then CSP does not receive I_{CSS} but both CSP and CSS receive a* resolution token *that also contains the SLA contract $\mathcal{S}$ between CSS and CSP.*
- *Timeliness: $\mathcal{A}$ terminates no later than the time T_E.*
- *Non-repudiation: When $\mathcal{A}$ terminates in (i), CSP is able to prove independently that the origin of I_{CSS} is CSS and CSS is able to prove independently the origin of I_{CSP} is CSP. If $\mathcal{A}$ terminates in (ii), both CSP and CSS can prove independently the origin and content of their* resolution token*.*

5.1.2 Proposed Architecture

After learning the SLA enforcement problem and other associated gaps in an abstract, a conceptual model was built as mentioned in the figure. This conceptual model, which is initially based upon a service buyer(CSS) and the service seller(CSP). These two participants form a two-party service exchange model, their service orientation, is configured to synchronous communication model with another key entity, the trusted third party(TTP) who monitors and mediates their service exchange. Each event is bounded in synchronous communication, therefore, responses are expected within a set time frame, before an alarm is raised. Further, our algorithm explains that how this architecture, handles the SLA enforcement by minimizing various challenges, security, and those events associated with compromising exchange fairness. Specification of each serving component such as SLA(E) and FE are also explained and their interaction with their other counterparts is also demonstrated their seamless interaction and responses accordingly.

The next ultimate task was, how this conceptual model is extended to a working tested. We implementing it into the cloud environments such as Microsoft Azure to see and record each entity's behaviour. The reason for this implementation was to analyze and verify how each node on the system model interacts when configured system specifications, operating environments, and in the presence of other constraints when we deploy CSS, CSP, and the TTP as virtual entities. Following, we share those details and figures, which were recorded to justify our findings, how this fair exchange-based in-line TTP

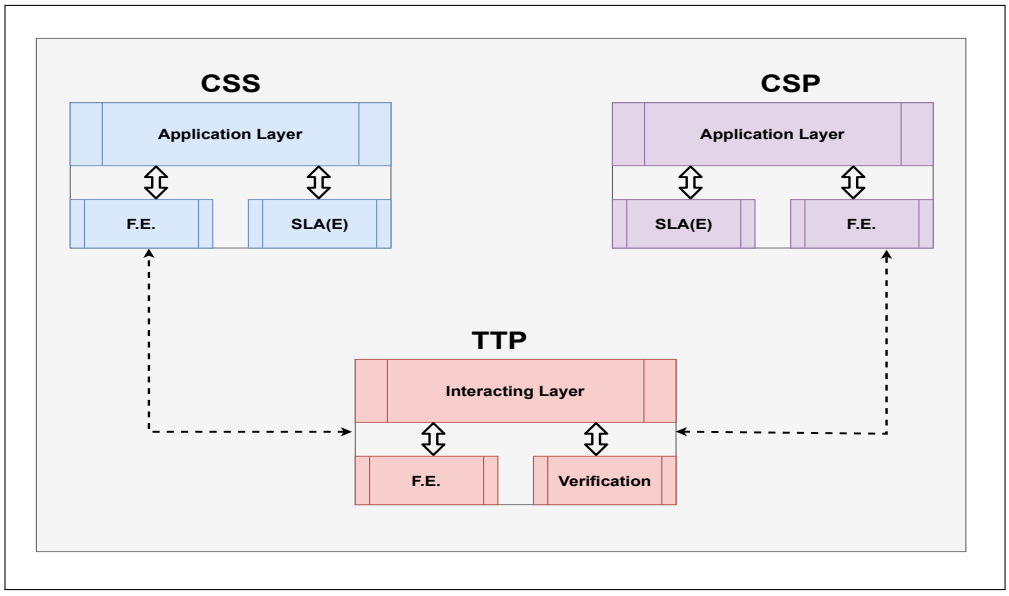


Figure 5.1: Proposed Architecture(I) *(when participants are loss averse)*

implementation handles the service exchange for ensuring a robust and resilient automated SLA enforcement.

Figure 5.1 shows the fair exchange component (i.e., a solution to the fair exchange problem) providing services to an SLA enforcement component. The idea is that the SLA enforcement component passes values or parameters to the fair exchange component. Once the fair exchange components receive the intended values, it executes the relevant subprotocol (i.e., normal, resolve or abort) by forwarding the messages to the intended party, i.e., the TTP, the CSP, or CSS. The fair exchange component at either the CSP or CSS is responsible for ensuring that the exchange has taken place. For example, the fair exchange component at CSS will ensure that the agreed set of sensors (i.e, I_{CSP}) is delivered but it will be the SLA enforcement module that will verify whether the sensor values (relayed by the fair exchange component) satisfy the agreed SLA.

However, the SLA enforcement problem is challenging because, typically, the CSS needs to "pay" upfront before the CSP provides services to the CSS. In the case of dispute, the CSS needs to convince the CSP that there has indeed been an SLA violation. Even in these cases, the CSP may disagree with the CSS for a variety of reasons, e.g., denying the source of the evidence (i.e., sensors). In such a case, the problem is the burden placed on the CSS. Further, a typical fair exchange solution is symmetric, i.e., both parties start the protocol at the same time by exchanging messages directly with each other, and that these message exchanges do not involve the TTP in a successful exchange or in a *normal* termination. On the other hand, the SLA problem is

not symmetric as the CSS needs to pay upfront for future services, and the CSP can return sensor values only towards the end of the time window for ongoing service provisions.

5.1.3 Architectural Components

Our proposed architecture has got three participants involved and users, which only causes the sensors at the CSP to be updated and do not participate in the SLP enforcement problem. The three participants are (i) CSP, (ii) CSS and (iii) TTP.

Table 5.1: Fair exchange protocol with post-exchange dispute resolution

Step	**Protocol**	**Messages from/to P_A**	**Messages from/to P_B**
1	setup	$\phi(A) = eK_A(I_A)$	$\phi(B) = eK_B(I_B)$
2	E	$M_A = eK_B(L, A, H(N), \phi_A)$: $P_A \rightarrow P_B$	$M_B = eK_A(L, B, H(N), \phi_B)$: $P_B \rightarrow P_A$
3	E	$Ack_A(B) = (L, A, H(N), H(M_B), my\_ack)$: $P_A \rightarrow P_B$	$Ack_B(A) = (L, B, H(N), H(M_A), my\_ack)$: $P_B \rightarrow P_A$
4	R	$Res_A = (L, A, H(N), M_A, M_B, \text{Resolve\_Req})$: $P_A \rightarrow TTP$	$Res_B = (L, B, H(N), M_B, M_A, \text{Resolve\_Req})$: $P_B \rightarrow TTP$
5	A	$Req_A = (L, A, H(N), M_A, \text{Abort\_Request})$: $P_A \rightarrow TTP$	$Req_B = (L, B, H(N), M_B, \text{Abort\_Request})$ $P_B \rightarrow TTP$
6	R	$M_{TTP}(A) = (L, TTP, H(N), M_B, my\_ack)$: $TTP \rightarrow P_A$	$M_{TTP}(B) = (L, TTP, H(N), M_A, my\_ack)$: $TTP \rightarrow P_B$
7	A	$Abort_{TTP}(A) = (L, TTP, H(N), \text{Abort\_granted}, A)$: $TTP \rightarrow P_A$	$Abort_{TTP}(B) = (L, TTP, H(N), \text{Abort\_granted}, B)$: $TTP \rightarrow P_B$

CSS and CSP

These two participants are those that require SLA enforcement. As such, there are two important components at each of these two sites: (i) A SLA enforcement (SLA-E) component, and (ii) a fair exchange (FE) component. The SLA-E component forwards or receives SLA-related information to and from the FE component, which is responsible for executing the fair exchange problem. Using the protocol of Table 5.1 as a template, we will present our SLA enforcement protocol in two ways: (i) information that flows between SLA-E and FE, and (ii) the messages that the FEs exchange between themselves.

Table 5.2: SLA Initialization

Step	Messages from SLA-E to FE	Messages from FE to SLA-E
1	$\psi_{CSS} = Sig_{CSS}(\mathcal{S}, V_{CSS})$ $\psi_{CSP} = Sig_{CSP}(\mathcal{S}, V_{CSP})$	L (see Table 5.3) L

Table 5.3: SLA Enforcement setup phase - Fair Exchange Component

Step	Protocol	Messages from/to CSS	Messages from/to CSP
2	setup	$M_{SLA} = (CSS, CSP, \psi_{CSS}) : CSS \rightarrow TTP$	$(CSP, CSS, \psi_{CSP}) : CSP \rightarrow TTP$
3	setup	$Sig_{TTP}(CSS, CSP, H(N), L, T_E) : TTP \rightarrow CSS$	$Sig_{TTP}(CSS, CSP, H(N), L, T_E) : TTP \rightarrow CSP$

TTP

Since the TTP is only responsible for enforcing the fair exchange problem, it does not have an SLA-E component. So, whenever the TTP is involved in message exchanges, it is the FE component that is responsible for the exchange.

5.1.4 SLA Enforcement Protocol

In this section, we explain the SLA enforcement protocol.

Initialization and Setup

We now explain the initialization protocol.
SLA-E, FE at CSS and CSP: Initially, both CSP and CSS need to send the agreed SLA contract $\mathcal{S}$ to the TTP, who can keep it in-store, in case of dispute resolution. To achieve this, the SLA enforcement component at CSS (resp. CSP) passes the following to the fair exchange component: (i) $\mathcal{S}$ and (ii) V_{CSS} (resp. V_{CSP}). The SLA enforcement component signs these values (see Table 5.2) and passes them on to the fair exchange component. When the fair exchange component receives these values, the fair exchange component executes the *setup* protocol of Table 5.3. It includes a nonce in the message, that uniquely identifies the current exchange "round". At the end of the execution, the fair exchange component receives a value L (step 1), which is a label that links to the SLA $\mathcal{S}$ at the TTP, and L is passed on to the SLA enforcement module.

FE at TTP: When the FE component TTP receives the messages from the (FE components) CSS and CSP, it checks whether the two SLAs agree. If they

Table 5.4: Message Exchange Between SLA-E and FE during Normal Exchange

Step	Messages from SLA-E to FE	Messages from FE to SLA-E
4	at CSS - $(L, TTP, \phi_{CSS} = Sig_{CSS}(I_{CSS}))$ at CSP - $(L, CSS, \phi_{CSP} = Sig_{CSP}(I_{CSP}))$	(L, ϕ_{CSP}) (L, ϕ_{CSS})
5	at CSS - (L, TTP, CSP, my_ack) at CSP - (L, CSS, my_ack)	(L, CSP, T_E, my_ack)

Table 5.5: Fair exchange protocol during normal exchange

Step	Messages from/to CSS	Messages from/to CSP
6	$M_{CSS} = eK_{TTP}(L, CSS, H(N), \phi_{CSS}) : CSS \rightarrow TTP$	$M_{CSP} = eK_{CSS}(L, CSP, H(N), \phi_{CSP}): CSP \rightarrow CSS$
7	$Ack_{CSS}(TTP) = (L, CSS, H(N), H(M_{CSP}), my\_ack) : CSS \rightarrow TTP$	$Ack_{TTP}(CSP) = (L, CSP, H(N), \phi_{CSS}, my\_ack) : TTP \rightarrow CSP$
8		$Ack_{CSP}(CSS) = (L, CSP, T_E, H(N), H(\phi_{CSS}), my\_ack) : CSP \rightarrow CSS$

do, then it returns a label L to both CSP and CSS that uniquely identifies the SLA. In any future communication with the TTP, that label L needs to be used to reference the SLA.

When the CSS and CSP receives the message from the TTP, their respective FE component passes on the label L, as it is the only SLA-related data but retains T_E, which is the time window during which the exchange needs to complete and was found in the SLA.

Successful Items Exchange

Table 5.6: Message Exchange Between SLA-E and FE during Dispute Resolution

Step	Messages from SLA-E to FE	Messages from FE to SLA-E
9	at CSS - $(L, TTP, CSP, \psi_{CSS})$	$(L, CSP, H(N), \mathcal{S})$
10	at CSP - $(L, CSS, H(N), \mathcal{S})$	at CSP - $(L, TTP, H(N), \psi_{CSP})$

Once the setup is complete, there is a round of items exchange. Typically, T_E may be a month, where the CSS renews its CSP subscription every month, but we are not concerned with the exact value in this paper. After the

Table 5.7: Fair Exchange Protocol during Dispute Resolution

Step	Messages from/to CSS	Messages from/to CSP
11	$Res_{CSS}(TTP) = (L, CSS, H(N), M_{CSS}, M_{CSP}, \text{Resolve\_Req})$: $CSS \rightarrow TTP$	$Res_{TTP}(CSP) = (L, TTP, H(N), \psi_{CSP}, \text{Resolve\_Req})$: $TTP \rightarrow CSP$
12		$Res_{CSP}(CSS) = (L, CSP, H(N), \psi_{CSP}, \text{Resolve\_Req})$: $CSP \rightarrow CSS$

initialization has taken place, for the CSS to use resources at CSP, he needs to submit his item I_{CSS} first, i.e., some form of "deposit". However, as mentioned earlier, the process is biased against the CSS. So, rather than sending I_{CSS} to CSP, he sends it to the TTP. Then, at some time before T_E, CSS will query the CSP to pull the sensor values so that it can decide whether the SLA has been satisfied. If the SLA is satisfied, then CSS informs TTP that SLA has been agreed and TTP releases the "payment" to CSP.

SLA-E, FE at CSS and CSP: Specifically, the SLA-E component at the CSS sends a signed "payment" to its fair exchange component (see Table 5.4, step 4). This is the item that the FE of CSS will exchange in return for the satisfactory sensor readings from CSP. The SLA-E at CSS passes the following to its FE component: (i) the label L, to reference the SLA, (ii) the participant to send the item to, in this case the TTP and (iii) a signed item. The FE component then sends this to the TTP (Table 5.5, step 6).

At some time before T_E expires, the SLA-E at CSP sends the signed sensor values (as agreed in the SLA) to its FE component, together with the id of the intended recipient (in this case, CSS - see Table 5.4, step 4). The FE component adds the relevant information, such as the label L and the nonce N to identify the current round and intended SLA (see Table 5.5, step 6). This is then relayed to the FE component of the CSS, which performs a first check on the validity of the sensors, i.e., whether the relevant set has been sent. If it is, the FE component forwards the signed sensor values ϕ_{CSP}, together with the label L, to the SLA-E component at the CSS (Table 5.4, step 4), who will be responsible for evaluating the sensor values and determine if there has been SLA violation. An important factor to note here is that, in case of SLA violation, a dispute resolution, if there is to be any, may take up to Δ time units to resolve. So, the CSP needs to transmit its sensor values to CSS before $T_E - \Delta$.

When the SLA has been satisfied, the SLA-E component at the CSS notifies its FE component of this by sending an *ack* (Table 5.4, step 5). The FE

component then notifies the TTP of this (Table 5.5, step 7) by also attaching a hashed copy of the sensor values. Following this success, the FE component at the CSP will receive an *ack* message from the TTP with the signed "payment" from the CSS attached (Table 5.5, step 7). This payment is then forwarded to the SLA-E component (Table 5.4, step 4), which then responds with an *ack* (Table 5.4, step 5); the FE component of CSP, in turn, responds to its counterpart in CSS with an *ack* (Table 5.5, step 8). SLA-E of CSS receives this *ack* in Step 5 of Table 5.4.

FE at TTP: During a successful exchange, the FE component of the TTP is involved in the following situations: (i) it receives the signed "payment" from the CSS (Table 5.5, step 6), (ii) it receives an acknowledgement from the CSS to proceed to pay the CSP for successful execution (Table 5.5, step 7), and (iii) for making the payment to the CSP on behalf of CSS (Table 5.5, step 7).

SLA Violation and Dispute Resolution

Whenever there has been an SLA violation, the notification from the CSS differs from that when there is a success.

SLA-E, FE at CSS and CSP: When there is a violation, the SLA-E component at the CSS sends a copy of the original SLA contract to its FE component (Table 5.6, step 9). This is then forwarded to the CSP, via the TTP (in step 11 of Table 5.7); FE of CSP forwards the Resolve_Req of CSS to SLA-E of CSP in step 10 of Table 5.6. The SLA-E component at the CSP then acknowledges this violation by sending a message (Table 5.6, step 10) to its FE component, which then acknowledges the CSS about this (Table 5.6, step 12).

FE at TTP: The TTP is involved in two ways during an SLA violation and dispute resolution: (i) It receives a message from the CSS informing it about a SLA violation, (ii) it verifies the validity of the CSS's complaint and, if valid, it notifies the CSP by sending a copy of the contract, rather than the "payment".

5.1.5 Correctness

Here, I prove the correctness of the SLA enforcement protocol. There are three properties to the SLA enforcement problem: (i) fairness, (ii) termination and (iii) non-repudiation.

Fairness:

There are two parts to this proof: (i) when the protocol terminates without a dispute resolution and (ii) when there is dispute resolution.

No dispute Resolution: When the exchange protocol starts, since the SLA-E component of each participant $P \in \{CSS, CSP\}$ transmits a signed item

I_P to its respective FE component, and since fair exchange satisfies fairness, fairness at the SLA-E level is guaranteed.

Dispute Resolution: A dispute resolution is triggered only if $\neg V_{CSP}(I_{CSP})$ at the CSS. Then, from the protocol, SLA-E at the CSS informs its FE component about the violation by sending a copy of the signed contract (Table 5.6, step 9). When TTP receives this notification from the CSS, it does not forward I_{CSS} to CSP, rather it sends ψ_{CSP}, with information about a dispute resolution. Since we assume a loss-averse participant, the CSP knows that there has indeed been a SLA violation.

Termination:

Successful or Normal exchange involves 4 communication steps: CSS receiving sensor readings I_{CSP} from CSP, CSS sending *ack* to TTP, TTP in turn passing I_{CSS} and *ack* to CSP, and CSS receiving an *ack* from CSP. If we ignore time taken for local computations and assume the bound on (synchronous) communication delays to be D, the normal exchange can terminate in $4D$ time after CSP has initiated its transmission of I_{CSP}. Note that CSP can resume its cycle $c+1$ services to the CSS while executing the last communication step in parallel.

Dispute resolution also completes in $4D$ time after CSP has initiated its transmission of I_{CSP}: after receiving I_{CSP}, CSS sends its *Resolve_Req* to TTP, TTP then resolves the request and informs CSP, and finally CSP informs CSS of an 'adjusted payment' for cycle $c+1$ services for SLA violations in cycle c. Thus, $\Delta = 4D$.

The CSS, after dispute resolution, can initiate a new fair-exchange execution with adjusted payment. This will involve only 3 communication steps for CSP to resume its cycle $c+1$ services: CSS transferring the adjusted payment (in Step 6 of Table 5.5) to TTP, sending its *ack* again to TTP (in Step 7 of Table 5.5) and TTP forwarding the payment and *ack* to CSP (in Step 7 of Table 5.5). Assuming that execution times are negligible compared to D, the first two steps will complete within D time; so, CSP should initiate its transmission of I_{CSP} for cycle c no later than $T_E - 6D$ so that the payment from CSS, even after any SLA violations, reaches CSP no later than T_E.

Non-repudiation:

Non-repudiation captures the fact that the CSP or CSS cannot dissociate themselves from some commitment. For example, the CSP cannot claim that the sensor values do not originate from it. Here, we need to show that the CSP can ascertain that the sensor values I_{CSP} originate from the CSP. In Table 5.4,

step 4, the SLA-E signs I_{CSS}, which is then sent to the FE component at the CSP. Since FE satisfies non-repudiation (see Chapter 3) and the FE at each participant (CSS and CSP) forwards the signed item obtained to their respective SLA-E component (Table 5.4, step 4), then SLA-E can guarantee non-repudiation.

We have proved that the SLA-E protocol, using the FE component, satisfies the SLA specification.

5.2 Protocol Implementation

Within distributed computing realm, the term protocol is described when communicating among system processes using a set of rules and formats to perform certain tasks. On top of other key formation elements of a protocol, both the sequence of exchange messages, the data format along their agreed specifications [52] play a pivotal role to form a protocol. Whereas, a communication protocol is *"an established set of conventions by which two computers or communication devices validate the format and contents of the messages exchanged"* OR *"a method by which two computers coordinate their communications"* [158]. In this section, we will be examining our proposed protocol in a more detailed fashion. We will discover our FEP *fair exchange protocol* based solution's*architecture*, *scope*, *limitations*, *properties*, *mechanism* and *flow controls paradigms*. We will also elaborate on the test environment, how various dotes can be connected to implement the design, components, and various challenges at different computing environments.

In its general construction, there have been some software, network, and traffic simulating modules were deployed to check how our protocol interacts to achieve an automated SLA enforcement within the cloud environment. This implementation also demonstrates how various processes and procedures are binds systematically to govern three core phases such as *message exchange*, *message abort* and eventually the transaction's *dispute resolution* [160].

This paper was primarily intended to transform our previously proposed conceptual model [160] to an implemented prototype, with a concise yet technically rich description of each component and its associated operating elements. The conceptualization of our idea was nothing else but a high-level depiction of how the protocol's operation is organized and its expected system's behaviour. However, when it comes to the actual implementation, it is usually comprised of the protocol's physical, logical, and service regimes to serve multiple participants e.g. in our case, the CSP, the CSS, and TTP.

Physical Regimes

Virtualization has transformed the computing service industry. Resource provisioning, deployment, maintenance, on-demand serviceability, and cost factors are all undoubtedly convincing. In the cloud computing perspective, physical computing entities refer to those virtualized computing resources which are provisioned to a service subscriber e.g. IaaS. Hardware which could include storage, networking, compute, load balancing, security-related technologies is either made virtualized or they could be offered as is so the cloud customers can gain direct access to the CSS data centers, by opting for a premium IaaS service.

Logical Regimes

In the client/ server architecture, a logical regime covers various processes, customized applications, and bespoke software to perform certain tasks for business reasons. These obligations could cover manipulating the data in various aspects such as to send/receive, process, compute, store or even redirect incoming service requests from a single or multiple clients to the designated serving nodes within distributed system environments e.g. cloud computing.

Such logically designed platforms are capable to incorporates expected interactions among different data/ security controllers and business processes to perform concurrent tasks. Data access, data exchange, batch processing, establishing intercommunication, network integration, or even termination of an entire service are some of the examples. These technologies also facilitate storing granular information for cross-references to another appliance or service module.

Services Regimes

The service scope defines such service‘s boundaries where business participants would agree to certain service obligations so RoI can be well justified. Outsourced cloud computing services at either CSS or CSP’s end ascertain operational controls, service dimensions, optics, deployments inclusions and exclusions taxonomy, service monitoring, service reporting [129]. It also states a fault‘s definition, ownership, fault-tolerance, rectification, and in the worst-case scenario service termination. Services scope would also equally dictates financial terms, conflict of interests, service duration, contractual renewals, dispute resolutions, and escalation procedures.

Participants

Figure 2 briefly shows how our four business participants are inter-connected to serve and deliver cloud-based services. This different architectural view enhances the knowledge to comprehend participant's computing estates in terms of operating scope. It also enlightens how some core modules are provisioned such as a simulated end users estate using *Apache JMeter*. The FEP controller integrated within *CSP*, *CSS* and the *TTP*. Master SLA repository is also provisioned for cross-referencing to these participants except the end-users as they don't need this module. SLA-E module would only be integrated into the CSP and CSS as TTP wouldn‘t require it either.

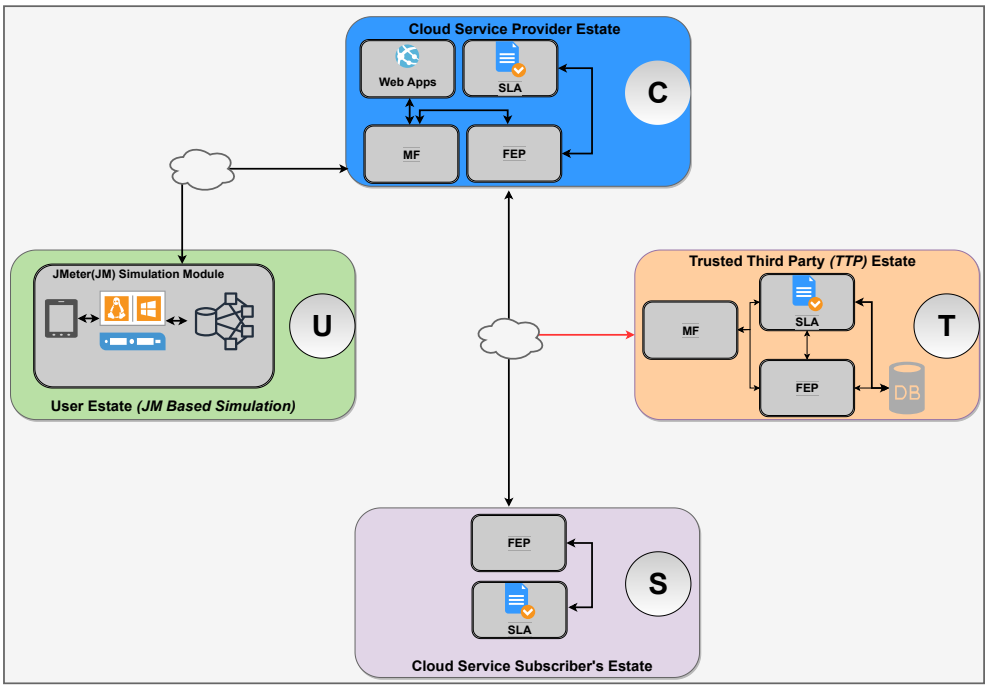


Figure 5.2: Architecture's Operational Interaction - Preview

5.2.1 Protocol Mechanisms

This section sheds light on how our protocol's elementary functionalities are woven together. Before the protocol initializes prime components *e.g. message exchange*, all those mandatory operating conditions *(e.g. SLOs, dispute resolution, service termination, etc.)* should be determined and satisfied along with their essential properties as mentioned in Chapter 3. Various service elements like how the protocol‘s sequence to be monitored? How message exchange is determined, when transaction participants for instance *CSP*, *CSS* & *TTP* would exchange digital services against agreed financial obligations. Defining how anomaly detection procedures are set in place, is another significant phe-

nomenon? The protocol‘s flow and error control are indeed paramount in a way to avoid unnecessary service overheads such as latency, performance, auditing which eventually could cause temperamental and intermittent SLA violations.

Protocol’s Sequence Control

Protocol sequence control is significantly important. This is linked to the messaging order how messages *(to/from)* are being sent and received followed by a proven logical sequence. This arrangement works through an algorithm that generates a unique message identification number *(UMIN)* reference, with none or least probability of producing an identical UMIN which could cause chaos while exchanging messages or serving requests to/from clients. These arrangements also incorporate a distinctive message exchange signature for a digital forensics tractability in terms of authenticity, correctness, completeness. The UMIN processes are often agreed upon among service providers and service subscribers to identify the message‘s authentication, reconciliation *transaction completeness* and auditability *(missing/ disputed transactions)* for tagging each message for bookkeeping purposes.

This exercise is also beneficial to set message classification in terms of their prioritization, agreed by one or perhaps all participants. In our test implementation, where we are simulating users estate as *(U)*, messages are systematically generated by Apache JMeter, which facilitates the definition of a UMIN e.g. thread reference number or an e-Tag is wrapped around every single message is exchanged back and forth. MF which records *(Request In (RI) and Request Out(RO)* after time-stamping each message. It eventually forwards each message to the web service hosting facility. Similarly, it receives the response from the web service hosted by the CSP and repeats the same process be recording the RI and RO, before it sends the message response to the originating entity U. This entire sequence data can be preserved for further investigation to understand the data provenance when it was communicated among different entities if and only if a dispute resolution scenario arises.

Protocol‘s Flow Control

This feature administers the chocking off state when an enormous amount of data is swamped through from one computing entity *(The Client)* to another *(The Server)* causes extreme service bottlenecks. The responding node, therefore, has to put some kind of flow controls in place to avoid such a state when their request handling and processing capabilities get stranded. This is mainly achieved by critically monitoring and measuring the incoming/ outgoing traffic, outsourcing the number of process/ query / compute handlers, and

determining dedicated resources before getting into a denial of service state. Flow control sensors mainly capture traffic information such as what *layer 4 protocol* is being used, *source IP* , *source port* , *destination IP*, *destination port*, these values are also known as *5-tuple* [3]. Flow control is also managed through message transmission-blocking and non-blocking conditions, depending upon the resource provisioning, business expectations, and service deployment orientation. Following are some of the states, which further explains message flow control:-

- *Time Oriented Flow Control*: This control refers to a dedicated time frame agreed between the CSP and the CSS. For instance, SLAs do yield what kind of computing tasks can be performed either in a peak time or off-peak time. These time bared services are designated to arrange smooth traffic flows to and from the CSP data centers. Subscribers are encouraged to stick within these allocated time frames so to minimize unfair service allocation and resource availability to other subscribers. Most of the time CSP's are also privileged to either *reset*, *release* or even *terminate* service connection at any stage, if their flow control monitoring sensors report some abnormalities. Flow control is also conditional upon what financial commitments are being made between the CSP and the CSS. Provisioning a cloud service with regards to SLA is also measured using the time orientation e.g. AWS Amazon.

- *Performance Oriented Flow Control*: Service performance is one of the key element of a cloud service. It counts multiple factors, when evaluating a service performance e.g. bandwidth, response time, latency and availability are some of the major service elements when it comes to the service performance.

- *Security Oriented Flow Control*: This element of protocol flow control is responsible to ensure security mechanism. Traffic flow is critically audited from service security perspective. Message exchange is conditional to traffic pattern and behavior. Traffic data is periodically reviewed using signatures repository to understand how traffic flow is passing through the data centers. For instance Amazon AWS like other cloud service providers offers a monitoring feature where traffic flow logs of a virtual private cloud *VPC* can be captured and monitored. This feature monitors any unusual traffic flows occurred, due to a service faults, security compromise or a perhaps a service downgrade via *(request for configuration change)*.

Protocol's Error Control

In digital communication, a protocol‘s error detection Intrinsically depends upon other functionalities related to the message‘s integrity. It determines a seamless message exchange between two parties constituting if, sender‘s and receiver‘s authenticity, validation, verification elements, are reliable on unreliable networks for instance internet. Therefore, error handling is anticipated on higher priorities as it could modify a service state from success to failure. These errors ranging from failed tasks, communication, or processing time-outs or modified system-level permissions, could distort part or even the entire service delivery hence violating the SLA. Various schemes such as Checksums, Parity Checks, Cyclic Redundancy Check *CRC*-enabled appliances and devices can instantly analyze messages for an unexpected error, integrity, accidental or maliciously injected faults, or any other similar abnormality. A communication protocol in a distributed systems realm depends upon a multiple-layered error handling mechanism. The first message fails within the end user‘s environment and never leaves their network. The second message leaves the end-user estate but never arrives at the CSP estate. Finally, a state when CSP processes the requests but those requests for soem reason stay within CSP‘s network eventually hence arrives the end user‘s estate.

Let‘s briefly explore the AWS error handling techniques. They have mainly divided the errors into two categories. First is classed as Client‘s errors (error codes 4**), where AWS APIs doesn‘t accept the client‘s request due to some obvious reasons. This could refere to either authentication, verification or validation of either the credentials, parameters or agreed configurations. Second category is of those errors, which are associated to the AWS‘s service environment such as their networks, compute or storage related (error codes 5**), where these serving nodes either do not process the client‘s request(s) or if they do, the number of retries or request re-submissions to resolve, is too long. This unnecessary time to process those requests consequently times out, which causes violating the SLAs. Predominantly, a forensics analysis should be able to discover the failure causality, especially if occurs within their environment so to decide if service credits will be rewarded to the clients by the CSP, in case of a disputed service delivery. For instance cloud service like AWS Amazon, the *AmazonClientExceptions* and *AmazonServiceExceptions*, both hold potential information that where the service failure occurred [13].

5.2.2 The Architecture Scope

Architecture‘s scope covers fundamental service and operational obligations. Here author tries to illuminate some of operative expectations along with those

situational states when our protocol would behave incorrectly. As already mentioned we don‘t have to go through the individual component‘s working however some of their scope elements needs to be explained here. Participants must agree upon the protocol deployment‘s conditions *including perimeter-configurations at both Ⓒ & application level settings at Ⓤ* are met. The protocol‘s proof of concept will run to demonstrate it‘s authoritative and operational capabilities such as monitoring, aborting, halting or even a temporary termination *(in case of an unfair and an unsolicited service delivery)* transactions. Dispute resolution, the prime objective, will be triggered, when a dispute is marked as unresolved between the CSP and the CSS. Each participant knows their corresponding trust boundaries, service role and how the FEP module‘d integration will be collaborated.

Any sub-components at any participant‘s environment, which directly or indirectly, effects SLA enforcement or constraints service monitoring, till the end of the agreed term, would be considered as an auxiliary-module of this architecture. Assumptions will also be taken into the account e.g. *sub-protocols*, *processes*, *procedures*, *systems configurations* and *segregated service monitoring controls* are all successfully implemented by all the participants and the master configuration *SLA metrics* are duly recorded into the FEP master node at TTP. This is how FEP module, will be able to enforce an automated, flawless and trustworthy dispute resolution after verification, validation and persuasive collaboration with SLA-E modules at Ⓒ and Ⓢ. Retaining security logs, events and capturing SLA metrics *(such as response time, bandwidth, latency, etc.)* within their own environments, will also be considered as protocol‘s core functionality.

Reviewing *figure* 5.2, where although all participants *(C.U.T.S)* are independent computing entities, yet, they are somehow bound through mutually agreed FEP component, which synchronously evaluates SLA compliant message exchanges. Often CSPs do define some shared responsibilities for running, securing and operating their provisioned services. In SaaS, additional adversaries, related to the data at transit, *(insider/ outsider cyber attacks, malicious participants, intentional/ accidental service compromises, service degrading, etc.)*, between the end users estate and the service provider‘s trust boundaries, are out of the scope of this paper. These global topologies such as internet, involve abundant 3rd parties, acting as transitional hosting services *(mid-tier service carriers)* while transporting the data, could cause a serious service disruptions, such compromises do occur, however, those adversaries will be omitted at this stage.

JM‘s Scope

As this paper demonstrates a prototype, therefore, we chose to simulate the end user‘s estate using Apache‘s JMeter. The test plan is configured to depict a very basic *HTTP* traffic, which is directed to an assigned IP and port of a remotely based *MF*. JMeter‘s test plan configuration can be changed accordingly to show certain variance of users activities. For the sake of our this implementation, the basic requirement was to generate users inputs, who are trying to access a cloud service by just visiting the cite. JMeter‘s mandate is to either send or receive these messages to and from the intended final recipient *(e.g. cloud service)* via MF. JMeter‘s other obligations at it‘s application layer could be, to record other key metadata. JMeter would record HTTP‘s various response codes [34], depending the service negotiations, up time, availability and successful provisioning. It keep dispatching the messages to it‘s recipient until either client‘s given time slot is exhausted or one of service termination signal is generated locally for various reasons.

MF‘s Scope

The *MF* is only obliged to receive incoming messages, timestamps them and redirects those messages to the actual cloud service, where it‘s being hosted. That cloud service process the query, resend them to the MF. These responses are received by the MF, which again timestamps them and send them back to the end users. MF also calculates the response time and records the entire transmission, which periodically sent by the CSP to the CSS so the SLA compliant transactions can be marked before paying the upfront to the cloud service for the next term services.

We also require a message forwarder *(MF)* which forwards the signals to predefined destination node(s). FEP module as we learnt is the central brain which makes the QoS assessment with the notion of SLA enforcement. This module is deployed at CSP, CSS and the TTP’s end in a fashion where these modules are inter-linked and can establish the communication among these participants when a violation occurs or a dispute query arises.

FEP‘s Scope

FEP module is the central brain which makes QoS assessments with the notion of SLA enforcement. It analyze every single message against SLA‘s master configuration repository, while CSP‘s response transmissions are being exchanged. Those stats mainly holds message date, time, transaction id and the response time. On behlaf of CSP and CSS, MF runs an evaluation scheme which produces the ART. Based on these ART, the FEP sends the alert to the

TTP. Actions like Exchange, Abort, Resume and Terminate a communication channel also falls within FEP's scope.

SLA(E)'s Scope

SLA(E) not only collaborates the FEP to reconcile the SLA but it also refreshes the latest version of an updated SLA between CSP and the CSS whenever a new service negotiation takes place. It holds copy of the master configuration file *(stating the SLA metrics e.g. the agreed response time)*, duly approved and acknowledged by the TTP.

5.2.3 The Protocol Limitation

While designing & implementing a protocol, certain limitations and operating constraints cannot be ignored. These weakening elements could easily impact a service delivery in a distributed SOA architecture. Therefore, to pin point the failure, one has to understand all the service dimensions at both, the client's end and the provider's end. A slight variation of either computing resource allocation, changes in system configuration, adding unnecessary restriction to the system access and other putting additional security overheads, could cause a serious problem to a running system process by either terminating & disrupting or even producing falsify signals. Following is the list of the which can be considered as the most critical of those constraints, which could leave the impact behind:

- Assets Accessibility
- Metadata *(e.g. web service metrics)* file format
- Data Interporatability
- Timely Negotiations *(SLA & SLOs)*
- Conflicting Interfaceability
- Unpredictable Faults at Operational Boundaries
- Legal & Regulatory Constraints
- Monetary Limitations
- Clock Synchronization
- Unidentified Attackers & service compromises
- Involvement of another 3rd party/ service contractor

As a matter of fact, most of the cloud service providers are a bit stringy when it comes to monitor and collect metadata of provisioned services. Apparently, they do offer very restricted security surveillance of their web services as they do use various throttling techniques to minimize resource sharing. Amazon‘s CloudWatch and other 3rd party tools collaboration is out there, however, monitoring restrictions, such as what is being monitored? How much is being monitored? What is being collected and how much is being collected. Some of SaaS solutions, do allow to integrate their APIs with tools for specifically monitoring Application Performance Management *APM* like AppNeta, Dynatrace, and DataDog however to acquire them is far too risky for service subscribers to associate these 3rd parties. These entities can certainly monitor and collect, client‘s data and even these vendors cannot guarantee that subscriber‘s business requirements will be met or not to monitor a particular service element. Therefore, a service subscriber could end up provisioning too many different tools to meet business requirements on top of the CSP‘s additional charges to monitor and collect these metrics too. Our protocol does face some of these limitations at certain computing stages as some of the service elements are composed and managed by the 3rd parties when messages are being sent or received. Refine tuning of various service performance, scalability, and reliability considerations are reviewed and implemented to ensure a successful communication is established with a minimal failure probability.

Architecture‘s On-premise Virtualized Deployment-SN1 Architecture testing on a single machine would ultimately exhaust system resources, especially when multiple resource-hungry tasks such as a continuous stream of message exchange, are being processed. Single CPU would only handle certain instructions per process whereas project execution requires a platform with appropriate compute resources and a higher level of service resilience within a distributed environment such as virtualization. The following figure shows how we intend to test our FEP based architecture in different environments.

Virtualization is the prime *software* technology that serves the global cloud computing environments. Implementing these miraculous frameworks empowers the existing limited computing resources so they can be transformed into multi-shared computing infrastructures for various business purposes and service provisions. An abstraction layer sits between the application and the underlying computer hardware, which adds more flexibility, reliability, security by diminishing the redundancy and economic factors at the same time. For getting advantage of these embedded features, we commissioned few virtual machines.

We placed all our prime transaction participants on dedicated virtual

machines and established their interaction through setting up endpoint communication channels and system permissions and corresponding configurations. This arrangement facilitates JM, which will be sending simulated messages to another virtual machine, representing an MF, which redirects the network traffic to a preconfigured endpoint, the webserver. This web server acts as the CSP‘s node, where we have configured an application server so to respond to the original client's request and vise versa. We configure MF on each node except TTP. The MF module on CSS primarily performed a critical task by evaluating the required SLA metric *ART*. Time stamping and recording each HTTP request simulated by going through JM back and forth through MF updates a metadata repository. As previously agreed, the designated CSS node would send SLA metrics as a *(.csv file format)*, periodically to the CSP for reconciliation and forensics purposes. These periodic transmissions would establish the fact whenever the SLA gets violated so the affected participant can be compensated through the fair exchange protocol. QoS also gets evaluated so if the said service is found SLA compliant then only the next service term will be agreed upon by paying the upfront fees, otherwise, the service will be terminated.

While running this experience which consists of few dedicated on-premise virtual machines, the expected ART indicates a latency due to the system resources do get overstretched and eventually it shows anomalous alerts of ART, when calculated. Through this exercise, we also noticed that we do get an unusual and unstable response time, which was probably because of overloading the system processes and delays. Concurrent resource utilization gradually reduces the CPU and memory allocation and provisioning for upcoming and awaiting threads sent by JM. Although the number of simulated users we set on our JM test plan was as per the recommendations by Apache‘s guidance and other best practices advice, however, the local machine‘s resource handling causes a bottleneck for the VMs which is hosting the sample web application to responds to messages effectively and efficiently.

Architecture‘s Cloud Deployment-SN2 The commercial emergence of cloud computing has been a phenomenal revolution and great motivation to the business world. The migration of corporate information assets such as infrastructure, development platforms *DevOps* and web based services, holds amazing attractions for enterprises of all sizes, however, it introduces some critical challenges too [188].

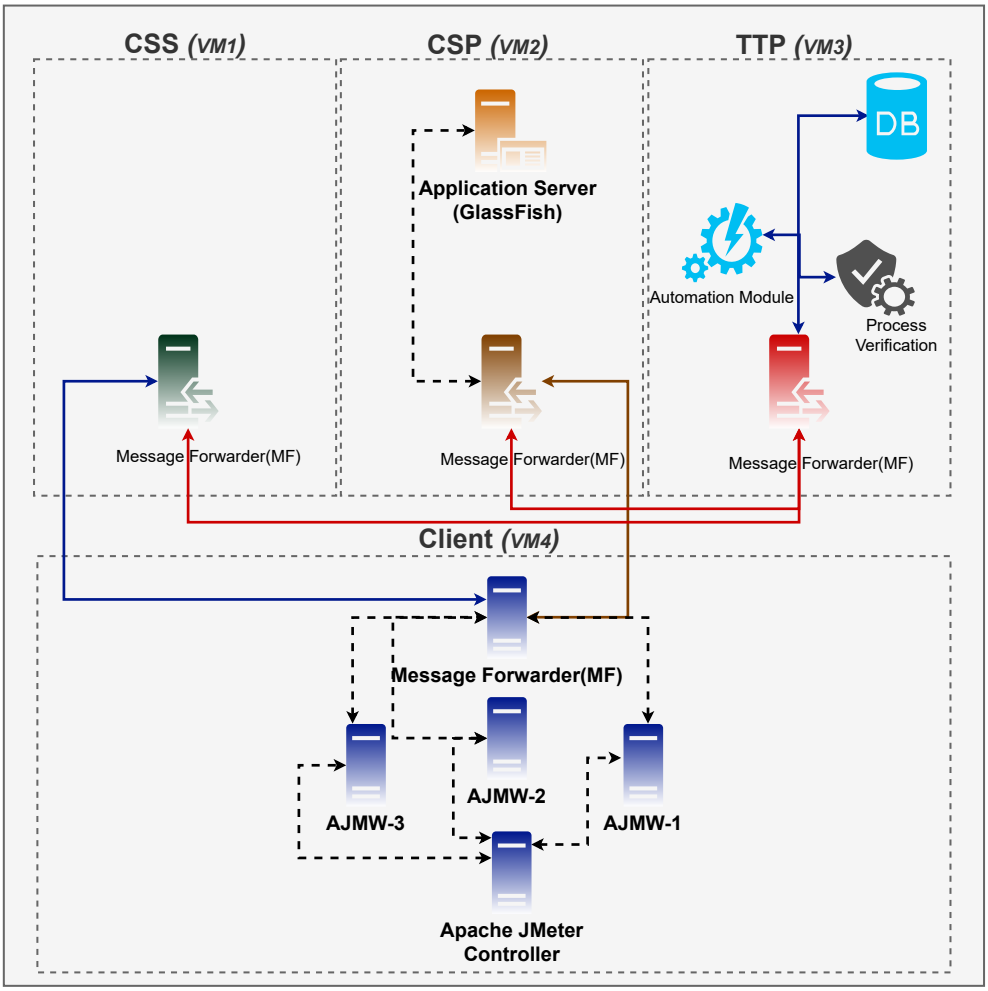


Figure 5.3: Architecture's Cloud Deployment on Azure

The following Fig 5.4 illustrates the architecture in terms of defined processes on each node. Technical processes that the protocol may carry out, depending upon the what phase and what state the participants are in, at that specific point. This further refers to various stages like Exchange, Abort, Resolve stages. During the service setup/ initialization phase, most of the operating terms are negotiated so to ensure timely process execution and expected interaction at each node. This concept where these four protocol stakeholders, carry out their message exchange whereas TTP being the service monitor and who sits in line evaluates these activities between CSS and CSP. Any anomalous behaviour is triggered via pre-defined processes such as FE and SLA(E) as described in the above interactions gets artifacts verification against the master SLA, stored at TTP's SLA repository which might be holding other SLAs for other clients. The entire service and process execution are bounded via full-time synchronization. FEP server is mainly residual at TTP which keeps interacting that how the service exchange processes are being evolved

throughout the service cycle till the end of normal or a reported abnormal termination, which rigorously performs the reconciliation against the master SLA, dispatched and acknowledged by both the CSP and the CSS. Thus these interconnected and fully synchronized processes ensure an end to end fairness.

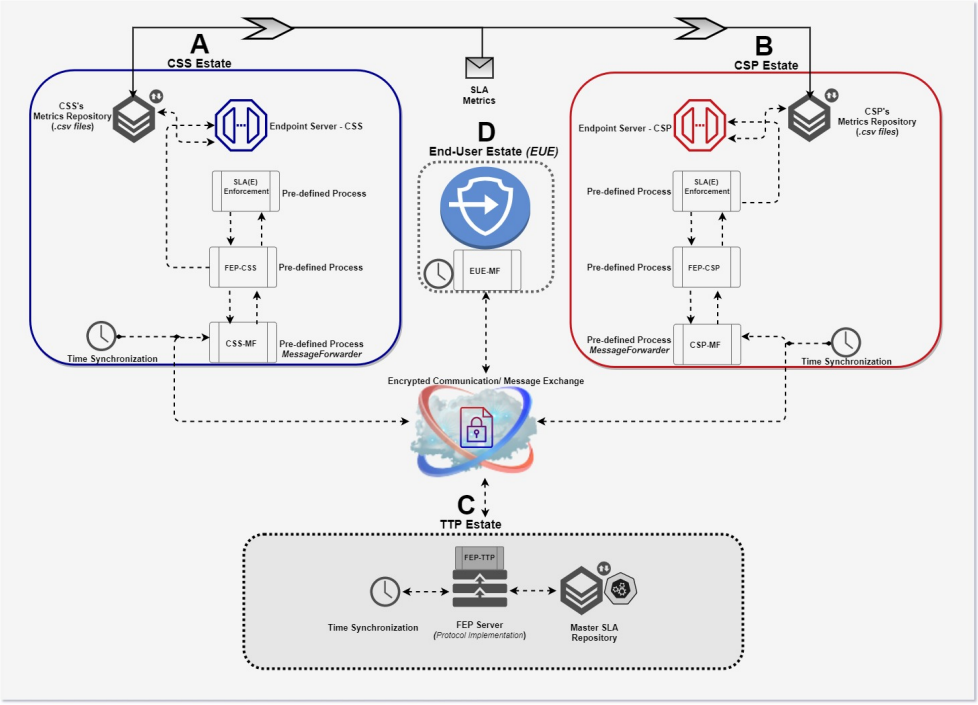


Figure 5.4: Process Execution

Generally, there are three main classifications of these challenges such as technical, operational, security [5, 82]. The scope of these challenges includes resource interoperability, granularity, service orientation, commercial factors, security, performance, QoS. All these factors are bundled in a mutually signed service level agreement for it's corresponding service or set of services, we have carried out some discussion about cloud computing(CC) & SLA Implications [74] and their associated key service elements and attributes. The scope of our testing is to demonstrate how do we achieve SLA enforcement by employing fair exchange protocol at the final phase of our test-bed.

Among multiple cloud service providers *e.g.* Microsoft Azure, AWS Amazon, and Google, we chose Microsoft Azure as our prime test-bed because of its popularity, flexibility, economical factor, and the number of other features offered by the service provider [6, 115]. Azure compute instance or a virtual machine, is an elastic compute segment. These resources are featured with instant cloud provisioning, where their subscribers do get full freedom to select various customer's centric cloud services and Cloud Application Programming

Interfaces *CAPIs*. This includes lower-level cloud infrastructure deployments to a full enterprise-level application and other strategic automation solutions. Unlike an on-premise computing deployment, most CSPs do offer minimal configuration, maximized scalability, rapid set-up time, secure and robust failure resilience platforms.

To mimic a real-world cloud service scenario, we assume that the CSS *A* provisions some web service to their client estate*(D)*, which was originally leased from the CSP *(B)*, against some financial commitments by setting an SLA. As previously learned that the SLA shows the entire service delivery plan such as deployment, SLOs, QoS, list of expected services, and the compensation workouts, in case if the CSP fails to deliver those promised SLOs within an agreed amount of time or an agreed scale of resource availability.

Our foremast focus is on ensuring the precision of our results especially when our protocol's administrative handlers such as an *Exchange*, *Abort* and *Resolve* gets triggered, acknowledged, and reported to and from the designated participants. Therefore, we tested our protocol on numerous cloud deployment schemes, tools, techniques, and procedures. It surely helps to refine our participating back-end operating nodes like JMeter, MF, and other monitoring and response sensors. This would also better align and evaluate our architecture's system configuration, inter-communication schemes, security composition, and fault tolerance to mimics a real-time SLA-based cloud service delivery with the least performance issues and other potential technical bottlenecks so a mutually trusted autonomous SLA enforcement can be fictionalized.

CSS Environment

To depict our participant's estate, we require to create four virtual instances for representing CSS, CSP, TTP, and the client environment. Although this was previously tested on Amazon AWS and other cloud resources, however, the last testing was performed on the Microsoft Azure Cloud environment. Explaining our architecture deployment, let‘s refer our readers to the Section 5.2 so defining our infrastructure instances with their corresponding logical and service orientations, capabilities, and functionalities. For the CSS node, as Azure offers a variety of OS‘s e.g. *(Windows or Linux based)* images, we opted an instance series which is a canonical, Ubuntu 18.04LTS VM with 2vcpus, 8GiB memory, and 30GiB standard SSD. In terms of CPU, memory, and disk input/output operations per second *(IOPS)* utilization, our experiment‘s initial requirement, and estimated usage were not projected as huge, therefore, provisioning some premium resources, was unnecessary at the moment. An Azure VM instance offers the users the flexibility to deploy multiple instances, start, stop, restart or even terminate them whenever requires. Because the CSS

instance represents a central hosting server, where some strategic programs and monitoring sensors have configured, therefore, it will be useful if these segments can be first discussed.

First, discussing our Java-based MF application, which resides here. It does not only proxies the message stream between the CSP and the Client estate but it also records key metrics *average response time*, in our case. It also timestamps every single (in/out)message, which could later be used for reconciliation purposes, if and when needed. A successful service exchange is the prime administrative control of this protocol, of which bases the SLA will be enforced. In a synchronous system, the messages are sent in a methodical sequential arrangement when a client is either seeking a server response over HTTP/ HTTPS. This message exchange is usually organized either in a synchronous *blocking* or an asynchronous model *non-blocking*. End users at the client's entity D would seek some of the server's responses for example one of the HTTP status code either *2xx Success,3xx Redirection, 4xx Client or 5xx Server* from the intended web service, provisioned by the CSP. We assume that at the beginning when the CSP service is delivered as a proof of concept, the client tests their accessibility and gets the HTTP status code 200 for an OK or pingable service. Depending upon the number of users at the client-side, a message which is holding some metadata about an HTTP/HTTPS request would probably include some requests from client to server and consequently some responses from the hosting server to the intended client. The same MF agent also holds the metrics which are bundled into a .csv file format and stores them into the Azure Storage Blob future experiment to gain and test more agility and resilience while testing our architecture.

Our future work will be carried out based upon an asynchronous communication model but for now, because the scope of this work is woven around synchronous communication, we are not using any of the robust message queueing systems designed and well approved for message queuing based upon asynchronous such as *RabbitMQ*, *ActiveMQ*. We will be using *Pika* which works best for both the asynchronous and synchronous models.

To create this environment, Microsoft Azure is the best option as they offer a huge service catalog to chose multiple elastic services among other vendors. They facilitate for instant provisioning of multi-purposed cloud-based systems hosting to serve as a compute, store, network, and other business-critical services with less hassle. To depict our participant's estate, we require to create four virtual instances for representing CSS, CSP, TTP, and the client environment. Explaining our architecture deployment, let's refer our readers to Fig. 5.3, where we categorized our three-fold deployment consisting of

the infrastructure estate, depicts the physical virtual machine(s) pointing our architecture‘s various participants. Then it comes to the logical deployment scheme shows how do they inter-communicate and finally how the service fabric is knitted around it all.

For shortlisting our cloud experiment *(virtual)*infrastructure, we chose an Azure compute instance or a hosting model where we can have a full end-to-end administrative control of participating nodes within our testing environment. These services could be containerized using Azure compute service catalogue such as Azure Kubernetes Service *(AKS)* or application service aka *App Service*, however, we, at this stage, chose to run independent virtual machines for each participant. For the CSS node, Azure offers a variety of OS‘s e.g. *(Windows or Linux based)* images, we opted for an instance series which is a canonical, Ubuntu 18.04LTS VM with 2vcpus, 8GiB memory, and 30GiB standard SSD. In terms of CPU, memory, and disk input/output operations per second *(IOPS)* utilization, our experiment‘s initial requirement, and estimated usage were not projected as huge, therefore, provisioning some premium resources, was unnecessary at the moment. Because an Azure VM instance in fact is a virtual machine so it offers the users the flexibility to deploy multiple instances, start, stop, restart or even terminate them whenever it requires. Another fact is that for our test implementation we opted the free-tier version of Azure instance, which put a lot of usability and maintenance restrictions. The vendor offers a certain number of hours per instance per month to run as free with no or very minimal support however for heavily usage the enterprise versions are available with better features and higher resources specs *e.g* 24x7 support, back-up, monitoring, and other corporate maintenance privileges.

An additional component such as configuring a virtual network, network interface*(NIC)* settings and network security group *(NSG)* to define allowed or denied state for our network traffic is implemented with accordance to our requirements followed by the service provider‘s guidance and best practice.

Secure remote logins to these nodes, protocols like SSH *(Secure Shell)* or RDP *(Remote Desktop)* are used once obtaining RSA public-private key pair along with the corresponding IP address of the intended node. Access keys can be downloaded to access the instance remotely using above mentioned remote access protocols. The VM instance is also associated with public and private IP addresses for connectivity and identification purposes.

The reason we prefer Ubuntu on Microsoft‘s Azure environment was due to its popularity, agility, scalability, and seamlessness. No wonder, due to these features most of the development environments and innovation platforms do prefer to run Linux on Azure or other similar cloud computing estates. That‘s

why this central server, is built using an Ubuntu server image for its suitability and compatibility with our protocol implementation on this prototype. Being the CSS central node, this controller node is capable of hosting most of our workload processing and monitoring the cloud services been provisioned to CSS's multiple client estates, however, for this experiment our scope would be to monitor a single client and their provisioned cloud services in terms of how CSP delivers the QoS against the SLA and how our suggested architecture diligently and methodically enforces it.

This is the instance where our Java-based module MF is also being placed. This would eliminate any unnecessary delays and other network tribulation and minimize the deployment overheads. MF receives the traffic coming from the user's estate and forwards it to the cloud services. It is also capable of returning the signals to its initialing node within the user's estate.

Application Deployment To build our application estate, *AEB* works miraculously to deploy scalable web applications and other similar cloud-based services. Because of its diversified deployment platform, enterprise-level applications developed using java, python, go, etc. can be deployed using wide-ranging application server environments such as Apache, IIS, Glassfish, etc. in a pretty straightforward manner. The AEB automatically handles all kinds of challenges from capacity provisioning to load balancing and multi-featured health monitoring along with rapid alerts distribution to multiple recipient platforms e.g. email, SMS, etc. Application deployments are also equipped with auto-scaling triggers to match the resources utilization to meet business and financial limitations [14].

CSS Environment

The CSS environment *(S)* holds the FEP module and a copy of SLA. FEP module here is not being depicted as the signals from the user's environment "U" are directly forwarded to the AWS EC2 instance, the CSS estate, however, getting the feed from the CSP's FEP. Furthermore, using this feed, an alert can be configured which can send generate a signal when the SLA gets violated. For this purpose, monitoring services like *AWS CloudWatch, DataDog, Dynatrace* can be used by adding pre-defined SLA metrics. This monitoring can be further extended to compare and measure when and how the SLA was breached. Monitoring solutions can send prioritized alerts e.g. emails or text messages to the appropriate recipients and mobile devices respectively. These alerts can perhaps also feed another entity like Security Incidents & Event Management *SIEM* appliances like *Splunk, ArcSight*, or *QRadar* for further investigation and distribution. Moreover, for dealing with certain alert services like *Zabbix*, *RHQ* and *fluentd* are well known for their features to serve the purpose.

TTP Environment

TTP environment *(T)* is the entity that administers any disputes or service ambiguities, which could occur due to many reasons. In our deployment solution, we assume the TTP is deployed on a cloud instance with a FEP module. It also holds the copy of the SLA agreed between CSP and the CSS. Furthermore, it also hosts some other solution for verification and validation of messages among these entities to avoid any malicious traffic gets injected or other similar attacks like Man-in-the-Middle *(MITM)* or Man-in-the-Cloud *(MITC)* attack scenarios. To distinguish this entity's prime existence we suggest some kind of cluster coordination service that can maintain the integrity of node's configuration information. In our suggested implementation such a service would act as a centralized system that holds the SLA configuration and other system configurations within a distributed environment, With this goal in mind, we could deploy *Apache's Zookeeper, doozerd, etcd, consul, chef*, or *puppet*, to govern and publish the system configuration and other SLO related information on every single node while a periodic yet synchronized convergence is transmitted.

End Users Environment

In our implementation plan, the end-user production environment U plays a vibrant role. This entity uses the cloud services and acts as a full workspace to gauge the cloud service‘s performance capabilities reflecting the SLA. Here the user‘s environment can be depicted by an Ubuntu or a Microsoft Windows computing platform, comprises JMeter [18] which is Apaches‘s load testing simulation solution. Despite having other comparable solutions like *Taurus, SIEGE, LOIC*, and *BURP* JMeter helps to run multiple pre-configured load testing plans and analysis patterns. Its low-resource consumption feature, measures the targeted system performance and its operational behavior illustrating different communication protocols e.g. *HTTP*, *SOAP*, *LDAP*, *POP3/SMTP* and other similar regimes. For the sake of this paper‘s scope, we will be testing a sample web application on AEB environment.

Due to Jmeter‘s flexibility and other distinct features, it became the first choice to simulate the user‘s environment. This open-source tool contains both the GUI and non-GUI initiatives along with its platform-independence attribute, to both stand-alone and distributed test executions. While running the implementation, our JMeter test plan holds the entire information about the targeted server which holds the sample web application packaged under a .jar file format. Although, there are other multiple choices available on the table to deploy the subject web application either on a stand-alone application

server such as Oracle, IBM, VM, Microsoft, Apache or to host it via some automated application deployment resource such as AEB or a similar SaaS.

Jmeter in returns fetches various metrics like timestamp, elapsed time, response code, thread name, latency, transaction success, and failure message when an HTTP request sampler is composed. It produces a comma-separated value *.csv* file as explained in *RFC4180*.

5.2.4 FEP Implementation Requirements

There are certain pre-requisites which are considered mandatory prior to execute the protocol. These requirements act as individually however their robust interaction with each other helps to draw precise results how the automated SLA enforcement and dispute resolution is being carried out and monitored.

- A valid SLA between CSP & CSS
- A User estate
- A CSP estate
- A CSS estate
- A TTP
- A Message Forwarding Module
- FE & SLA enforcement modules
- A Secure communication environment with pre-defined validation & verification arrangements among all the entities

Additional requirements such as log analysis and other discovery tools can be opted depending upon the business requirements however as per the paper scope, this would be extended for future research work.

Protocol's SLA Negotiation

Every time a cloud service is provisioned, it is contracted under a valid SLA between participants such as the service provider and the service subscriber. Measuring a cloud service is only meant for both, the CSS and the CSP, so to attain acceptable fairness how the said service being delivered with acceptable QoS and other associated measuring capabilities & metrics e.g. *(from high level e.g. availability, bandwidth, average transactions per second, latency to low-level e.g. CPU, memory, etc.)*. Service scope is clearly defined along with deployment limitations and a potential service state [62]. Furthermore, for

QoS monitoring the business time is defined as Mon-Fri, 8:30 am - 6:00 pm as service *peak time* whereas the rest of the days and hours including public holidays are classed as *off-peak* time e.g. Sat-Sun. While putting these layers, the cloud service is presumably secure and reliable within predefined confidence bound, as stated by the service provider.

Table 5.8: SLA Sample Metrics

SLA Metrics	**Targeted Threshold (peak-time)**
Availability	$\geqq$ 99.999 %
Average Response Time	$\leqq$ 50 milliseconds

It draws a line between CSS requirements and CSP capabilities when a service delivery will be classified as *"service failure"* or *"SLA violation"*.

SLA Definition To put our test in action, we require a sample SLA so to configure our monitors and other sensors to observe the QoS exceptions while these entities *C.U.T.S* interact with each other in both stateful and stateless, cloud-based SOA architectures. SLA metrics evaluate four principle rules of a service request such as *(when start, where start, when end, where end)* at both the client and the server ends [61]. Out of a long list of various cloud metrics, we only chose an average response time *ART* to gauge the SLA. We understand that the TCP-based traffic is bound under a basic flow control mechanism and hence other operating factors could seriously impact achieving the desired SLO's in a distributed computing system such as cloud computing.

In real-world service provisioning, the elapsed time yields, the request execution by the requesting client node, till the requested service request is completed or the given set of tasks are fully processed by the serving node. This refers to the service performance monitoring so to compare whether the service is SLA compliant or not. Factors like poor bandwidth, an excessive number of users, longer processing time, and request complexity could affect the response time. A longer response time would ultimately pose a greater risk for SLA violation For our test environment 50 ms *(milliseconds)* is considered as the SLA threshold for a typical response time over the internet when one can expect the completion of a request [105]. ART can be calculated by dividing the net violations in terms of time outs by the total number of requests *generated by JM* multiplied by 100.

ART = total violation / total requests x 100

Protocol's Exchange Signal

As mentioned in the conceptual model section 5.2.1, we assume that the SLA has already been signed between the CSP *"C"* and the CSS *"S"*. The copy of the same SLA has already been sent by both these parties to the TTP *"T"*. The reason TTP keeps the copy of this SLA is entirely for future reference and will be directed it when a dispute situation arises upon SLA violation. Although the end-users are utilizing the cloud service, an alert is sent periodically by the C to the S, whose FEP module evaluates the SLA compliance by comparing the ART. The time when SLA $\mathcal{S}$ is transmitted to intended recipients it can also be packaged with other information such as V_{CSS} (resp. V_{CSP}) via Jmeter.

Protocol's Dispute Signal

Upon a successful initialization and setup of the protocol implementation by sending some test transmissions and their respective acknowledgments by C, S, and T, the BAU exchange cycle begins. We assume that S has already deposited its I_{CSS} to the T as a token, which will be released upon a satisfactory SLA compliant service delivery by C, the service provider. Meanwhile, from these BAU signal transmissions, we take 15 minutes of samples to depict the real environment to avoid huge data manipulation. For SLA compliant service delivery we consider a status-code 200 [203] which represents that a simple request has been succeeded within a qualifying period regardless of a method used e.g. GET method of HTTP *(Hypertext Transfer Protocol)* executed via JMeter by attaining our expected responses within specified SLA time e.g. 50*(ms)*.

Protocol's Abort Signal

To classify a service failure or SLA violation, should JMeter gets an HTTP status-code e.g. 4xx *bad request, unauthorized, forbidden or not found etc.* or 5xx *server errors e.g. service unavailable, gateway time out, etc.* it will be considered as a service interruption due to certain reasons as mentioned above. FEP module will evaluate when this situation occurs. The TTP is the entity which will eventually be contacted by S, to report the abort state. To further refine the scenario we use Jmeter‘s duration assertion which marks both *FALSE or TRUE)* responses. Here the one of our depicted trigger sent abort message, calculating the ART of each failed thread on periodic monitoring basis to the entity T. This shows an unnecessarily longer responses that is ART *($\geqq$ 50 milliseconds)* which indicates failed or non-SLA compliance service delivery.

Protocol's Resolve Signal

When and how these entities resolve the dispute: say our FEP and security modules would send their feed to the TTP, who would have enough evidence to decide a fair resolution considering the victimized entity using true SLA metrics & master copy of SLA. This will be performed after reviewing and evaluating service metrics at the end of the term. The malicious participant would not be able to defend such a state, hence, the dispute resolution benefit would be granted to the honest participant.

This is can be reviewed from a couple of iterations how the suggested protocol would initiate the resolved signal and how it would set the TTP and the participants to acknowledge it. See *row 4* and *row 6* of Table 5.1 and & also see *row 11,12* Table 5.7.

Post Reconciliation & Service Restoration

Once the outstanding disputes are set to resolve and the honest participants acknowledge the receipt of having their due service credits or financial dues fully paid, the FEP modules along with the SLE(E) would be set to a service term. Any potential changes to SLA/SLOs will certainly be considered so the SLA metrics are measured accordingly. Any previous poor QoS stats associating the causality of any SLA violations would be added to the service monitoring look-ups to avoid any repetition of the same SLA violations. Protocol sensors would be adjusted and the relevant information will also be cascaded to the concerns TAs and the TTP to avoid any false positives which could trigger an abnormal service termination or unsolicited configuration mismatch at either CSP or the CSS FEP modules.

5.3 Results

This section reviews results as an out come of our protocol‘s implementation. We tested about six schemes with different system configurations from local host *on-premise* to distributed computing models *e.g. AWS Amazon*, to depict real world business scenarios. Behind these optimization techniques, the key intention was to study how our suggested protocol operates *(in terms of service automation and SLA enforcement)* in different computing states. We chose ART as a sample QoS SLO metric, to gauge various responses while monitoring QoS factor of our sample web application. Placement of monitoring sensors has always been a critical task, therefore, our MF was provisioned separately as out of the end user trust boundary, on a dedicated appliance system. This would assist to obtain more refined and improved results. Initially, the role of a MF was only to redirect messages but later an extended mandate was also

given to act as a metrics monitoring sensor. It collects transaction stats and eventually pass them as per the agreed schedules. At the end of the monitoring, it produces a comma-separated values *.csv* file which is well known to be used on different platforms. Defining the SLO metric threshold is anther critical task so to set a realistic response time pointer. Shorter the response time would be better to to align the service consistency and other elements of QoS. These readings translate a response patterns from CSP to service endpoint within a distributed environment. After reviewing various web services response times, we decided to set the threshold of 50ms for the SLA compliant ART. Each message exchange would be responded within this time frame or else it would be classed as an SLA violation if the response time is greater to 50ms. We will briefly analyse how our captured SLA metrics can be used for the following four immediate tasks as diversified reconciliation:-

a) *Periodic SLA violations*
b) *Response Time variation*
c) *Net Message Exchanges*
d) *TTP's Involvement*

Reviewing Periodic SLA Violations

The first and the prime dimension to analyze the results is to have full robust visibility of when and how SLA violations occurred. This analysis presents concentrated results as we gathered from various experimental instances staged on our cloud-based implementation *e.g. Azure*. Refer to the Fig 5.3 the deployment scheme was configured considering an approach, which enabled us to focus each service node's benchmarking while message exchange was being performed. This focuses on our intended SLO metrics average response time *(ART)*. This deployment was tested on two types of Microsoft Azure virtual instances, first one was *a manually configured application server hosted on Ubuntu* where the other was an on Azure Elastic Web services *automated application deployment platform*. To capture these results, in a more refined manner, the said deployment was variously tested before probe the actual three days transactions at various times. This approach leads us to properly validate and verify the testing regime's real-time stability, accuracy, and precise capturing of the client/server responses. Another objective was to set the SLO benchmarking point which depicts a real-world business scenario. At this stage, one clarification is vital to specify about the placement of the sensor, where these results were being gathered. The chosen estate was an independent cloud entity where the MF node was configured, which does multi-tasking e.g. forwarding to/from messages as well recording their timestamps for auditing and forensics purposes in case of further escalation of dispute resolutions via legal holds.

Table 5.9: CSP's Captured SLA Metrics

SLA Metrics	Average Response Time(ms)	Agreed SLA(ms)
SLA-Metric1	43.78	50
SLA-Metric2	47.82	50
SLA-Metric3	49.85	50
SLA-Metric4	34.53	50
SLA-Metric5	38.24	50
SLA-Metric6	39.94	50
SLA-Metric7*	**68.09**	**50**
SLA-Metric8	39.07	50
SLA-Metric9*	**67.29**	**50**
SLA-Metric10	47.35	50
SLA-Metric11	47.58	50
SLA-Metric12	48.06	50
SLA-Metric13	48.43	50
SLA-Metric14	48.19	50
SLA-Metric15*	**100.06**	**50**
SLA-Metric16	40.45	50
SLA-Metric17	39.43	50
SLA-Metric18	38.84	50
SLA-Metric19	39.41	50
SLA-Metric20	44.61	50

These critical business stats will later be shared with the CSS periodically by the CSP as well the TTP when needed. Table 5.9 shows three columns, where the first column presents *20 sample transactions* the captured SLA-Metrics. The second column shows the service delivery's actual ART at calculated at the end of each transaction and the third column is the agreed and pre-configured SLA *(50ms)* threshold to be monitored by each FEP module sits on all three business participants. Each message exchange between the end-user estate and the CSP is evaluated against this threshold of 50ms. Monitoring an acceptable service request SLO metric *e.g. response time* against the CSP's SLA claim should be carefully tested prior to signing the SLA as urged by [8, 130, 168].

This table enlists about 20 different transactions, which were recorded within the CSP computing estate. These metrics are probed only for CSS as, both the service provider and service subscriber are mutually agreed to share these periodic samples on monthly for ensuring SLA compliance. The TTP for being an agreed monitoring authority would also get a copy of these records following a BAU *Business As Usual* procedure. All these participants would follow a standard agreed verification and validation process, prior to commence the message exchange. Our assumption is that the BAU message exchange would run for a certain period of time *(e.g. monthly or a longer service contract term)*. The provided sample represents just a fragment out of

those message streams mutually exchanged by the CSP and the CSS's client. The key determination was to constitute a legal and forensically proven digital evidence to support and settle any dispute between both the parties, if arises.

These numbers would also ultimately serve the auditing and reconciliation needs performed by pre-configured and automated FEP modules at their corresponding nodes. This short metric capture would only be held on a random basis while the service is being provisioned. An exception will be granted to those system states when the CSP's provisioned services are declared either as faulty or under an agreed/ pre-scheduled maintenance downtime. For the sake of our prototype, we merely recorded three days of the entire service month. Exchanging these metrics every day would not be feasible *(cost/resources)* but also be considered as an unnecessary overhead. It would also impact the service delivery by overstretching available computing infrastructure and human resources at both ends. Reviewing the Table 5.9, which highlights only three anomalies [service entries] *sla-metric7, sla-metrics9 and sla-metric15*, mentioned in red. As these entries yield the ART is greater than 50ms, so each of them would be marked as a potential SLA violation for that particular day/ time and transaction cycle. We also have to bear in mind that these 20 results are only three days worth of sample transactions which capture only periodic message exchange for sampling purpose.

Similarly, refer to Figure 5.5 which is the graphical illustration of the table above shows how the monitoring is being performed. It shows ART calculated with their associated log files which show SLA compliance however those three anomalies which are displayed above the SLA threshold line are those entries that didn't comply with the SLA as computed by the FEP module configured within CSS monitoring.

Reviewing Response Time Variation

Here we would analyze the response time variations which could occur for many reasons within the cloud computing environment. A response time is when a certain job is submitted to the serving node and when it gets completed, the time it takes is denoted as the *net response time*. Within our experiment, we measure this time for every single transaction however the SLA violation context will only be measured upon the ART for the entire specified period *(e.g. end of the day/ business (peak/off-peak) hours)*.

Figure 5.6 correlates the Table 5.10 which observes response time variation if the number of users is increased to meet CSS's business requirements, would ultimately affect the SLA response time by increasing it higher than the projected response time. Therefore, adding more requests simultaneously

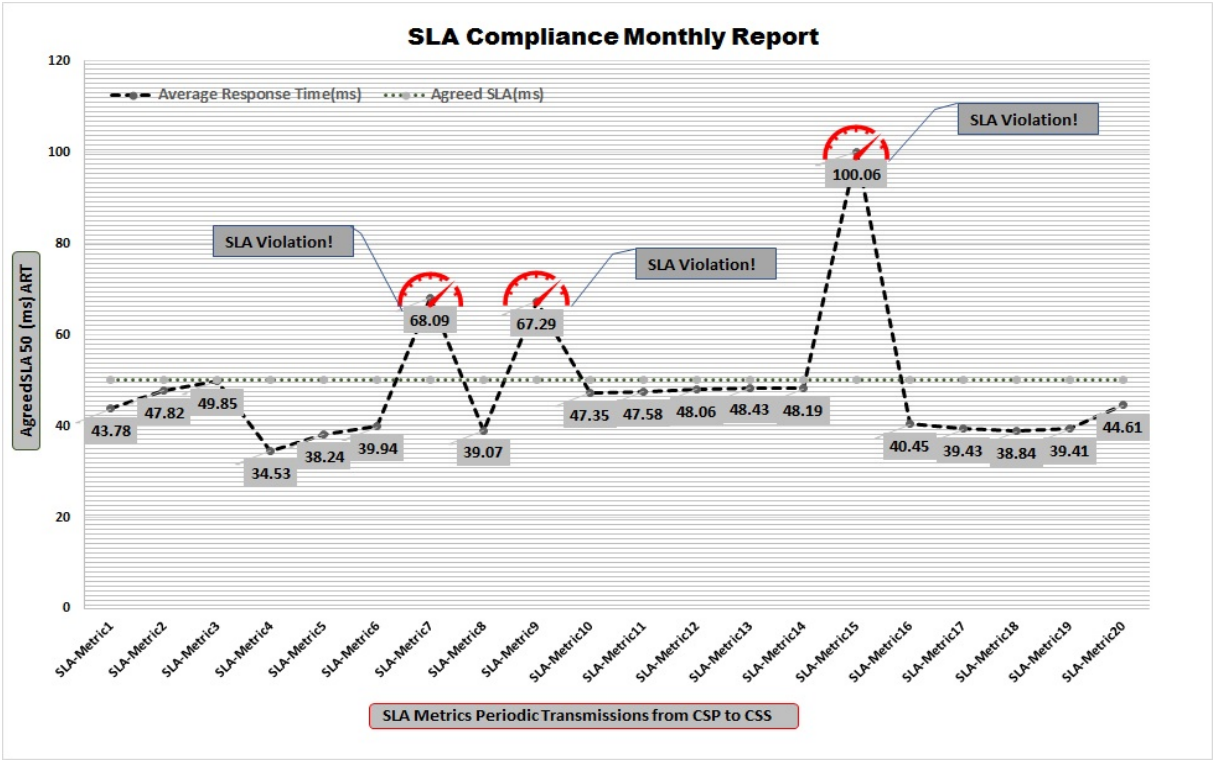


Figure 5.5: SLA Violations & Average Response Times

would result in a longer response time. This leads the situation towards SLA violation as the assertion duration can not be met until the SLA is altered accordingly.

Reviewing Net Message Exchanges

Fig 5.8 mimics the client‘s monitoring dashboard with the total number of transactions, regardless of their *(successful/ failed)* status throughout a service delivery day. We notice that those days when SLA violations were observed process a reasonable number of total transactions however these message exchanges don‘t fulfill the business requirements by responding to them within the agreed and graceful time/ QoS. Most of these HTTP requests are completed within a longer response time than expected. This observation constitutes rather a poor QoS sample. This interpretation trails a service situation where a lack of service quality without compromising the quantity of the transaction. The message exchange happens however the projected SLA is objectionable when financial commitments are concerned. We also drew a scenario from our test transmission that various occurrences under different time groups can be classified. Extending our SLA monitoring further, the instance 5.6 could also observe a particular time group when several requests were dealt with by the service provider in a peak or off-peak time.

Table 5.10: Response Time Variation

SLA Metrics	SLA Violation	Service Variation(ms)
SLA-Metric1	N	6.22
SLA-Metric2	N	2.18
SLA-Metric3	N	0.15
SLA-Metric4	N	15.47
SLA-Metric5	N	11.76
SLA-Metric6	N	10.06
SLA-Metric7*	**Y**	**+18.09 ↑**
SLA-Metric8	N	10.93
SLA-Metric9*	**Y**	**+17.29 ↑**
SLA-Metric10	N	2.65
SLA-Metric11	N	2.42
SLA-Metric12	N	1.94
SLA-Metric13	N	1.57
SLA-Metric14	N	1.81
SLA-Metric15*	**Y**	**+50.06 ↑**
SLA-Metric16	N	9.55
SLA-Metric17	N	10.57
SLA-Metric18	N	11.16
SLA-Metric19	N	10.59
SLA-Metric20	N	5.39

Reviewing TTP‘s Involvement

The TTP only gets involved when an anomaly is observed by active FEP modules. Initially, good service metrics correlating the actual SLA, are submitted to the TTP as a business obligation within our suggested protocol. This practice assists future reconciliation when settling any potential service disputes between participants.

This monitoring segment reviews those service delivery panics detected at TTP‘s end. As Fig 5.9 shows, three service anomalies are detected by the CSS‘s FEP module hence these anomalies are eventually acknowledged to the TTP‘s FEP module. TTP alerts the CSP‘s FEP module to verify and resolve these disputes by issuing a fair service credit or by repaying the equivalent payments.

The Fig 5.9 therefore, presents how often the TTP gets involved. During our test transmission TTP is involved about three times. The FEP modules at TTP would get these signals from CSS to examine these service compromises after eliminating any probabilities of having false positives. Upon every SLA violation signal, the TTP would execute an observatory module scripted with the following code:

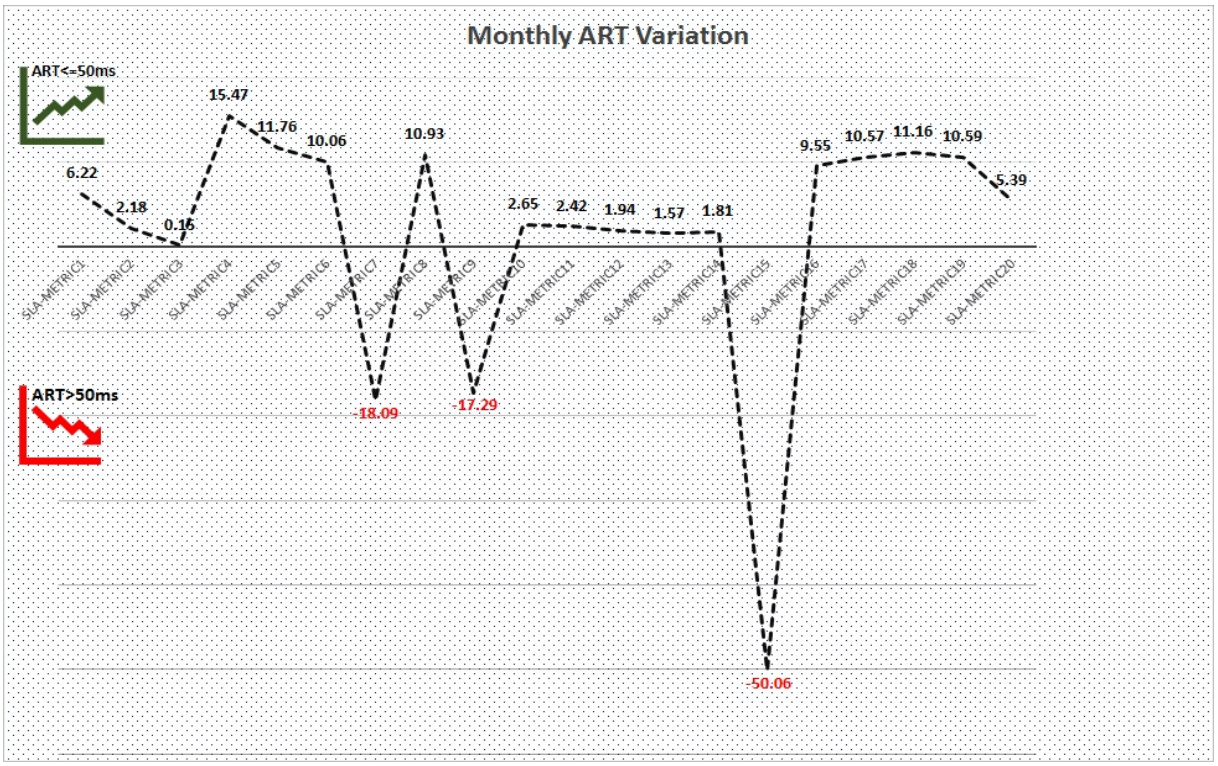


Figure 5.6: Service Response Time Variations

```
x >= 50ms
if x >= 50ms then
print "Warning!!!...SLA violation Detected"\\
or sends an alert via
SIEM (SMS or email)
else
print "Transactions are SLA compliant"
```

The module sitting at TTP with the above code would systematically manage the transactions and controls the dispute resolution by classifying with both, SLA COMPLIANT or NON−COMPLIANT messages.

In this chapter, we have provided a methodology for SLA enforcement. A small implementation was carried out to demonstrate how the prototype works under preset assumptions with variance operating and configuration settings. It does show the viability of our brief and restricted implementation. The test-bed also opens some of the useful avenues which will be helpful for further extending the implementation scope using versatile operating assumptions. Such modifications will certainly improve the protocol's functionalities for better and concise results achieving automated SLA enforcement.

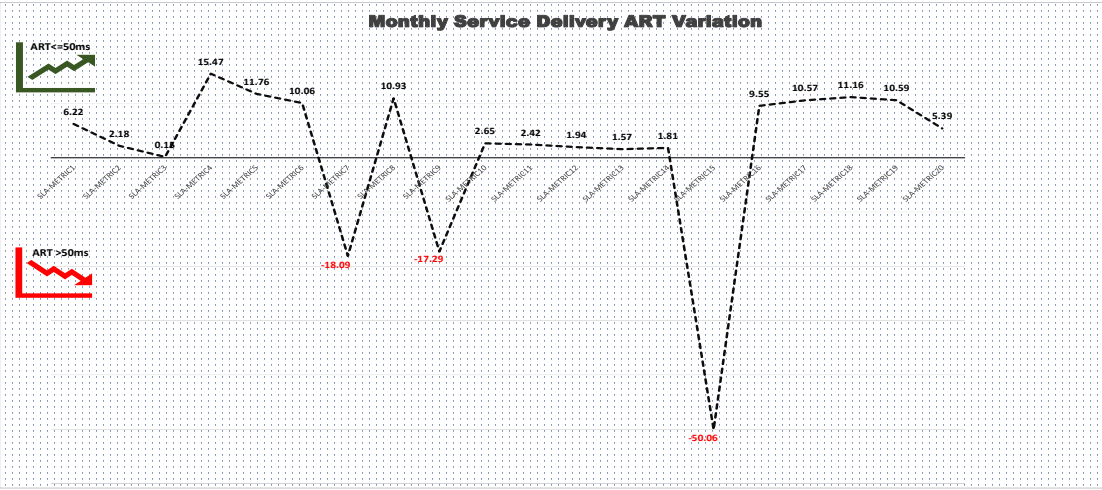


Figure 5.7: SLA Violations & Average Response Times

It also discovers how the implementation on the cloud environment and how underlying technologies can be trusted and connected to achieve data security, privacy while maintaining the automation perspective. This includes connecting various services and APIs so to check the possibilities of how the protocol would act. Certainly, this requires in-depth cloud deployment which is kept for future work. Despite all that, we were able to deploy our arrangement using a couple of cloud environments and were able to produce some useful and very encouraging results which depict how our protocol governs the service and the SLA management by not only monitoring or detecting but also enforce it.

The next chapter explores other protocol variance by evaluating and modeling, those possibilities when participants are intending to act maliciously. How their misbehaving intention can affect fairness at the end of the service term and what we proposed to mitigate those threat vectors.

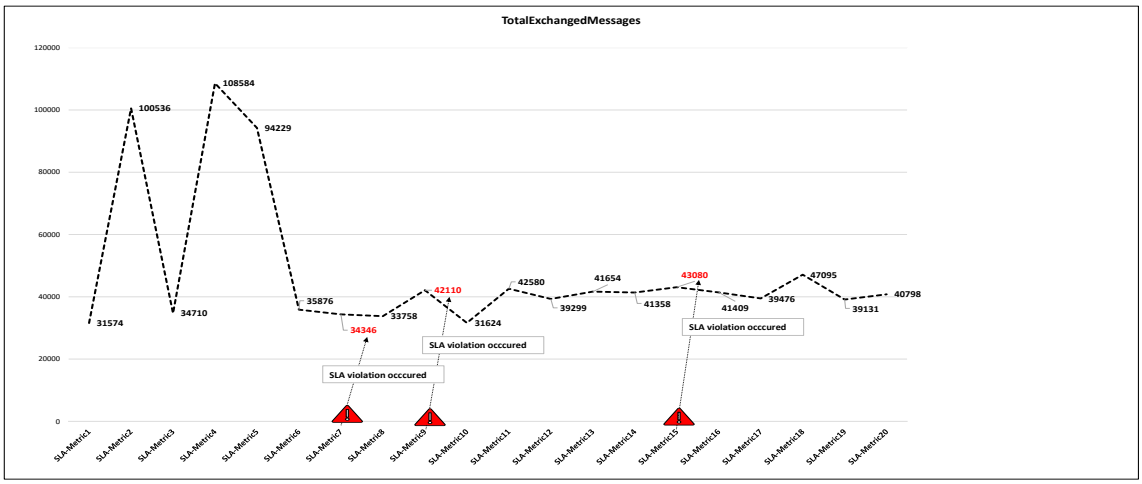


Figure 5.8: Total Exchanged Messages

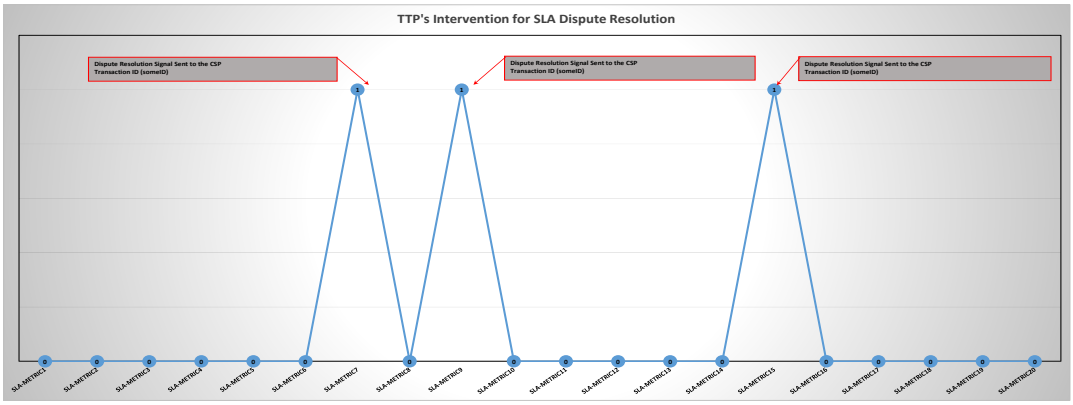


Figure 5.9: TTP Intervention for Dispute Resolution

Chapter 6

SLA Enforcement with Malicious Participants

The global technology industry heavily investing to ensure cloud migration by eliminating the legacy concept of conventional on-premise computing infrastructure to cloud computing spectrum, where various on-demand computing services are offered in a more convenient and resilient manner, serving end-users. This adaption trend still shows a huge number where multi-tiered cloud services are assisting businesses around the globe, regardless of their business size or type, by hosting their computing needs from simple web-based applications to complex distributed networks connecting B2B through the most innovative technologies. Since the global pandemic situation arose, where remote working has become a norm. Gartner's trend [191] shows how more and more global businesses are being attracted to switch their services to the cloud environments, which resolves the remote working issues by keeping their business liabilities significantly reduced. The said report forecasts that public cloud services to grow 6.3% in the year 2020. Among various cloud services, certainly, the SaaS is yet the most demanding service for assisting enterprises to embrace online office automation software suits by paying online subscriptions, which minimizes licensing implications, software assets maintainability, and their security elements.

Above computing trends, do bring some diverse challenges too for their subscribers who are registering their business interests towards these cloud technologies. As we learned previously in Chapter 2 about SLAs and their implications. Although, SLA being a prime legal contract between a CSP and the CSS, does enable both the participants accountable and provide a sense of protection, however, there are still a lot of associated challenges security issues, which could easily be exploited when a business entity decides to misbehave and be unfair with their other business counterpart. There are times when such

behaviours are not intentionally aiming for damage to someone. It could be a human error or an unpredicted machine fault or even a process failure. Despite SLA bindings, which provide guidance about SLA compliance, violation, accountability, dispute resolution, and lodging claim *(service credits/financial refunds)*, before their unresolved disputes reach out to the court of law for proceeding litigations.

In Chapter 5 a use case was reviewed when participating entities were loss averse and the suggested protocol depicted how automated SLA enforcement was achieved. Now If another scenario is discussed where participating entities demonstrate some anomalous behaviour, as such landscape is depicted in Fig.6.3. For instance, if the CSS believes that the CSP has not been provisioning their digital items/services as anticipated. Upon raising the concern the burden of proof is getting on the CSS's to satisfy their claims. Similarly, the CSP believes that CSS has not paid for its services. This is how during a service term either of the participants may decide to act maliciously. One option is to resolve mutually which is unlikely the case as the CSS won't have the capability to get low-level service metrics from CSP's environments. Another option is to escalates the case to the legal framework, again there are bright chances too that the burden of proof will be on the complainant. Modern legal frameworks, do appreciate and encourage to initiate some other alternatives to negotiate their service disputes out before the matter is escalated to the court. Other alternatives as discussed previously are mediators, arbitrators who can assist to resolve issues compensating and renegotiating for highlighted disputed SLA violations.

As a matter of fact, litigation or lawsuits can cost time and fortune in terms of financial commitments to a complainant. To resolve SLAs disputes, exploring some automated solutions would be very encouraged. That's how none of the participants may criticise such a dispute resolution mechanism. The prototype in Chapter 5 would not work for malicious participants. As CSS receives a periodic SLA summary report from CSP which helps them to reconcile their usage and most importantly the fact if the SLA has been violated or not in the past service term. Now to overcome this fear what if the CSP decides to tamper the contents of their periodic SLA summary file prior to dispatching it to the CSS. Their motivation could be to fake the bad-looking SLA metrics to cheat the CSS as if the SLA is meeting their expectations. CSS would not be able to detect such alterations.

Other potential threats may add more complexities and make this digital service trading platform a hostile environment that requires extended security and privacy measures. Previous architecture only supports an exchange environment where loss aversion is handled by involving an in-line TTP. When the

operating situation gets changed that protocol would require some ultimate modification to cope with the extent of emerging threats that could handle malicious participants.

As Chapter 5, discusses the protocol's various properties, the said protocol will be mainly working on weak fairness. Weak fairness holds such attributes which warrant gathering sufficient piece of (forensically sound) evidence using different security apertures such as secure coprocessors, fully homomorphic encryption modules, etc. Such arrangements can not only collect provable evidence but can also ensure a reliable dispute resolution functionality. Non-repudiation of origin and non-repudiation of receipt are the prime properties that satisfy Confidentiality Integrity and Availability aka as CIA triad.[59, 81, 169]

6.1 Architecture Threat Model

Threat modeling offers a methodical mechanism to evaluate any security weaknesses of existing or novel system architecture or a related process. It reveals where and how an attacker or a malicious business participant can exploit or misbehave to subvert security and the fairness elements of a machine, service, or process in place. A threat is, therefore, a signposting of those unprotected *(system or design)* fault(s) which potentially could cause some damage, so, threat modeling is the technique that maps those unseen threats so the relevant mitigation measures can be actioned. Threat modeling assists a team while reviewing the security around the project implementation towards spontaneous facts which could halt a service if a malicious actor *machine/human* could get involved at any *stage* using any *exploit* at any *time* [186]. That risky entity whether it could be a *service provider*, *service object*, *transaction component* or even a *supporting entity* with a capability to exploit via a threat agent. This could eventually lead to further privilege escalation into the targeted systems or service operations.

Previously, in Chapter 5, we presented a novel prototype with an operating environment that demonstrates a service exchange where prime participants are known as lose averse. Now using a new operating dimension and applying a proactive approach, we do protocol's wargaming, which depicts certain adversarial scenarios. This will let us review and discuss a participant's malicious intents and their associated mitigation steps. During the protocol run, in either case, one or both protocol's participants, if decide to act maliciously, what are the options how our protocol would interact the situation to take over and off course remedial arrangements. For instance what P_A, upon the agreed round completion would achieve. To gauge that situation, we assume

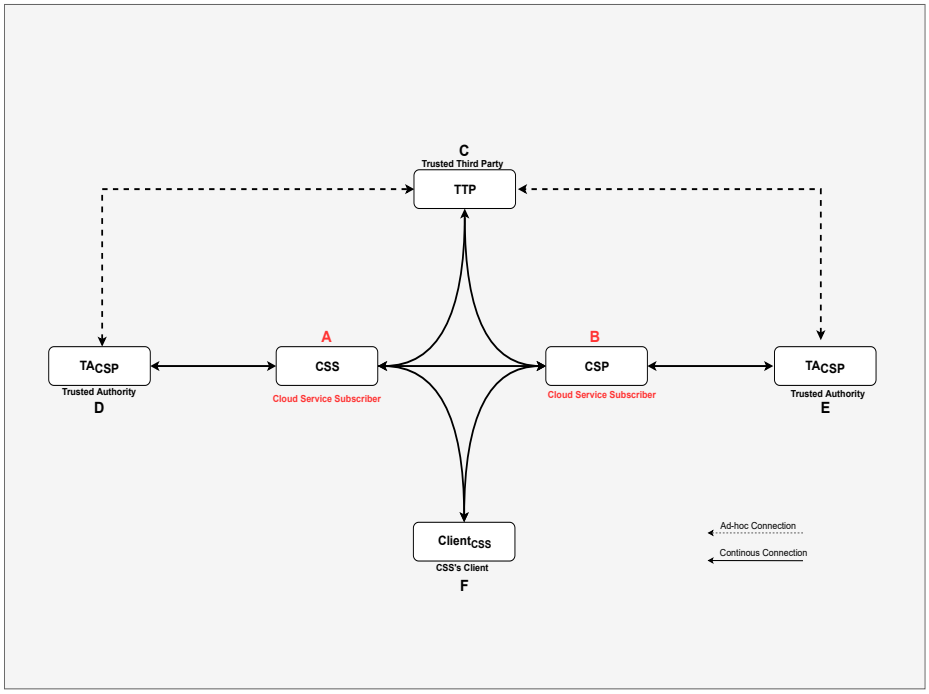


Figure 6.1: Cloud Service Exchange Basic Model

the protocol would have an end with either one of the following four eventual states, denoted as *St*.

- **St(1,1)** constitutes a successful exchange, where it has received both M_B and $Ack_B(A)$. It would further facilitate to computes I_B from M_B for having both the keys K_A and K_B.
- **St(1,0)** a state where this participant has received M_B but not $Ack_B(A)$, If it is in SA(1,0), it asks the TTP to resolve the exchange by sending message Res_A that contains both M_A and M_B. (See row 4 of Table 1.)
- **St(0,0)** where it has received neither,
- **St(0,1)** where it has received only $Ack_B(A)$. ,PA requests the TTP to abort the exchange by sending message Req_A that contains M_A."

6.1.1 Basic Service Exchange Model - *Potential Threats*

Refer to Fig 6.1, which depicts a generic cloud service exchange model, where six diverse transaction participants, methodically interact with each other. Both the CSP and the CSS, sub-delegate their trusted authorities (TAs) to

Table 6.1: Participants Interaction during Basic Service Exchange/round completion

Steps	Participant Sends	Participant Receives	Message Exchanged	Fairness	TTP Inc.	TA Inc
1	A	B	Payment	N	N	N
2	B	A	Ack	N	N	N
3	A,B	TTP	Ack(TTP)	N	Y	N
4	TTP	A,B	Ack(SLA)	N	Y	N
5	B	A	CloudService	N	N	N
6	B	A	SLASummary	N	N	N
7	A	B	ServiceTerminates	N	N	N
8	B	A	ServiceTerminates	N	N	N

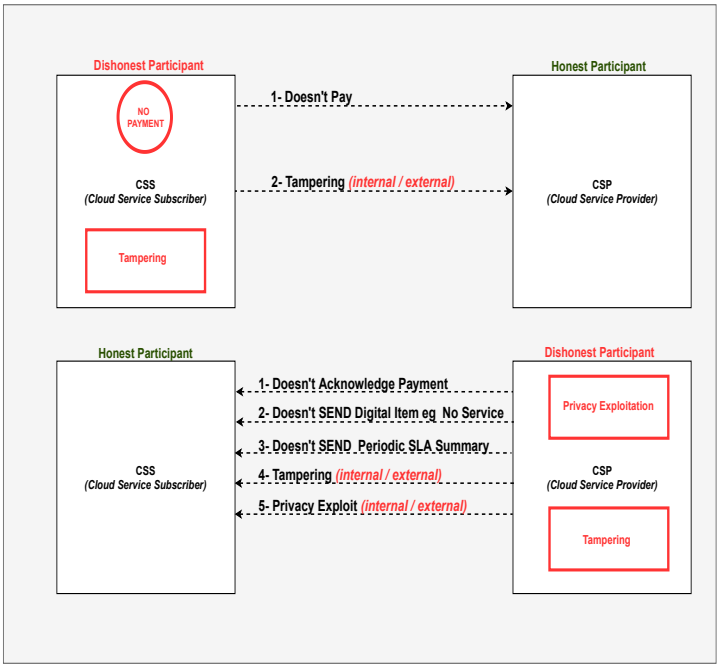


Figure 6.2: Potential Threats from CSP to CSS & from CSS to CSP

perform verifications and offer guarantees on behalf of CSS and CSP, when and where needed or upon receiving some instructions from TTP itself.

These verifications correspond to cover their specific items for instance a certificate authority (CA) would cover CSP's by acting as their TA, whereas, a financial clearinghouse such as a bank, represents the CSS instead, ensuring payment guarantees on their behalf, if and when TTP, mediates any disputes between them. A critical review of the service exchange model discovers few threats on both sides of the communication channels and within their corresponding trust boundaries. This demands to advocate some best solutions which can mitigate those threats so both participants can have an intuitive fair exchange by perceiving a dispute resolution mechanism based upon manually assisted (using trusted parties) or fully automatically (using some smart secure technologies) architectures.

6.1.2 When CSS Acts Maliciously

CSS-T1: Being a malicious participant, the CSS can either decide to stop/ refuse their payment(s), even though CSP fully provisioned their cloud services. A CSS may choose not paying them at all, pay an incorrect amount, pay late, or could demonstrate a similar unexpected behaviour which deprives the CSP to get their due payments. For instance, the CSS shows their grievance that SLA was only met for 48 weeks within the (52 weeks) service term, perhaps CSS may have observed some indicators of SLA violations (such as poor QoS, being provisioned with an incorrect digital service which doesn't match the agreed product specification and their business requirements as mentioned in SLA), therefore, accusing the CSP. Paying inadequately or incorrectly would still be considered in the same terms as not paying at all. Another similar, scenario where CSS could attempt to cheat by getting the items(service credits /financial refunds) twice by (a) using the cloud service provisioned by the CSP and claiming they didn't get it so, that's how the CSS might try to have both, by escalating the issue through the TTP.

CSS-M1: This threat can be mitigated by seeking an assisted dispute resolution through trusted parties such as TTP and corresponding TAs. Regardless of any of the above CSS's deception states (mentioned in T1), if the CSP doesn't get their due payment, the case can eventually be escalated to the TTP, who will review the CSP's claim and performs its due verification and reconciliation using their copy of the SLA and rest of the message contents furnished by the victimized party (e.g. the CSP here), which was initially shared by exchange participants, at the time of exchange set-up/ initialization phase. It develops a fork situation:

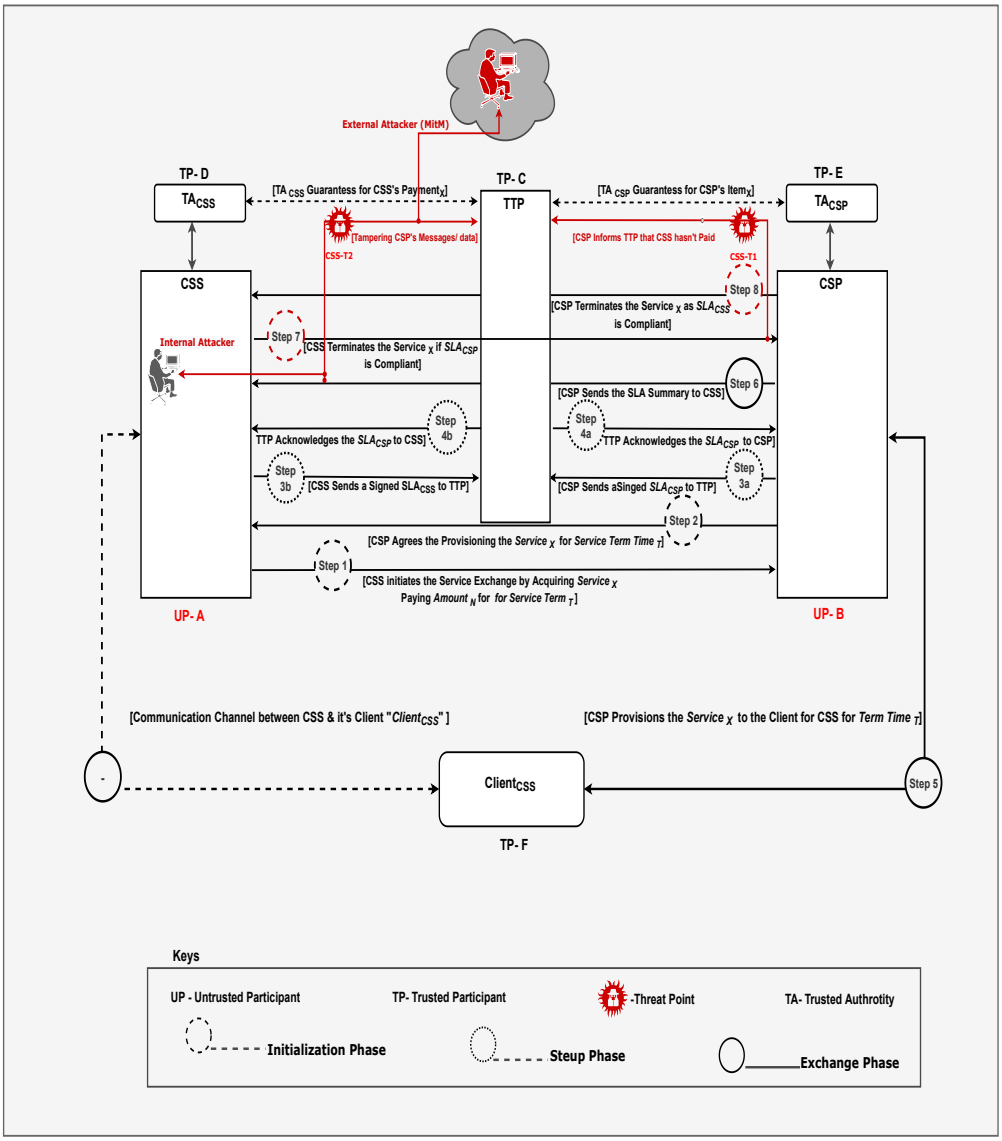


Figure 6.3: CSS Threat Points

- *Use Case (a)*: If the CSS claim is true that SLA was factually violated (e.g. between week 48-52) during the service term, the TTP, therefore, would instruct the CSP to compensate the CSS to resolve the matter, declaring the CSS being honest here. The CSS can get their due compensation (eg. service credits/ payment refunds, etc.) via TTP, forwarded by TA_{CSP} based upon a standing instruction issued by the CSP) and the matter will get resolved and the threat is confronted in this way. Furthermore, CSS cannot get both the item from the CSP and the claim the credit as well for the same item specification. This claim will be reconciled and verified at the TTP with the collaboration of TA_{CSP} who would verify that either the service item was fully utilized or the service credit was

issued.

- *Use Case (b)*: Emerges with a scenario, where the CSP is now proven to be honest and CSS tries to cheat the exchange means the SLA violation cannot be established through TTP's reconciliation and verification checks. This would lead to a state, where TTP, issues a set of instructions to the TA_{CSS} to generate the payment item(s) (as the CSS gave them TA_{CSS} this mandate at the beginning of the exchange to furnish financial guarantees, when needed), the correct payment item will be generated and will be sent to the CSP, which can resolve the dispute and mitigate this part of the threat.

CSS-T2 CSS or an Attacker tampers the Contents *(data/files)* During the service exchange, the CSP will have to send a periodic (monthly, quarterly) SLA summary to the CSS, which warrants final settlement payments to the CSP, *if and only if*, during the full-service term there was no SLA violation. A couple of scenarios arise here when CSP sends their SLA summary to the CSS, either the data gets modified during the flight halfway through, or it's intentionally modified upon landing on the CSS's computing estate by one of their insiders.

- *Use Case (a)*: The CSS is well aware the fact, that the payment can be stopped if the SLA summary proves that during the service term, there has been few SLA violations. Taking this note into the consideration, a malicious CSS can attempt to fabricate the summary report using various sophisticated *(anti-forensic)* tools & techniques to avoid any detection. CSS can then claim their case stating they will not pay because the SLA summary they received, clearly shows service was not SLA compliant and CSS eventually will escalate the matter to the TTP, who will review it against the SLA and will ask for the refund to be issued to the CSS through the TA_{CSP}.

- *Use Case (b)*: [MitM/MiTC tampers the contents] Now, another threat materializes when again the CSS gets a pre-modified version of the SLA summary sent to them by the CSP through untrusted networks. MiTM (Man-in-the-Middle) or MiTC (Man-in-the-Cloud) attack compromises the channel, the attacker modified the data and sends to the CSS, impersonating the contents are generated by the CSP avoiding any sort of attack detection.

CSS-M2: : This situation can be dealt, by using anti-tampering and privacy preservation techniques, preventing not only exchange participants but those

attackers who could gain unauthorized access through the untrusted channels. If a joint security protection regime is implemented on untrusted channels and within participants corresponding computing estate, this threat can easily be mitigated. Data on transit *(untrusted communication links)* and data at rest (participants computing estate) are somehow prone to various vulnerabilities and *(physical/ logical)* security attacks, which can be exploited by attacking entities (whichever insiders and/or outsiders) the targeted exchange platforms. Adapting Fully Homomorphic Encryption *(FHE)* can provision information privacy issues which obstructs by eliminating the chances of MitM *(Man-in-the-Middle)* & MiTC *(Man-in-the-Cloud)* attacks [102]. Extending this protection by also implementing the Security Modules such as Secure Coprocessors technologies *(IBM secure coprocessors)*, would ultimately prevent the insiders to perform any sort of tampering with the data, hence, eradicating insider attacks.

6.1.3 When CSP Acts Maliciously

While reviewing potential threats comparing CSP and CSS, one can tell that CSP being the infrastructure and service producer, can cause multiple adversaries to compare to CSS. A CSP, with a malicious intention, holds more business reasons, technical means, and logical implications to be unfair with one or more service subscribers on their tenant's stack. Refer to the threats landscape presented Fig. 6.4, where, a CSP could pose many threats to the CSS such as forfeiting their financial items, unsolicited modification of their service provisioning, data security, privacy preservation elements. Following details these threat elements and suggests potential mitigations.

CSP-T1: Assuming the fact that CSS is the entity within this exchange frame, who has to pay in advance, even before their intended digital services are provisioned on their desired platform. A CSP gets the payment at the set-up phase. Now when the CSS has paid upfront to the CSS whereas their expected service provisioned has not yet been carried out by the CSP, a malicious CSP could deny CSS's payment. Within the modern-day digital economy, where both the buyer and seller are anonymous, yet buyers do make payments to sellers first.

CSP-M1: To deal with this threat, a well collaborated integration of trusted third party along with corresponding trusted authorities representing buyer and seller, resolves the problem here. A TTP can be approached by the buyer when their expected items/ services are not delivered as per the seller's contractual obligation. TTP works as an arbitrator with more powers to prevent these participants to formally approach the regional legal framework to seek the

resolution. TTP, once the fact is established who the culprit is, informs them to resolve the matter, in this case, the CSP would be instructed to either arrange a payment refund or deliver the promised service at CSS's platform as they both initially agreed. As suggested, to distribute the segregation of duties, each business entity also has sub-contracted TA, who would guarantee on their client's behalf, to regenerate their items (either digital service or financial items), upon receiving TTP's instruction, to do so. This arrangement would relieve participants while agreeing to an exchange set-up where they can confidently negotiate any unexpected post-exchange disputes and their corresponding resolutions. It eventually gains trustworthiness, between, CSS and the CSP, by maintaining their full anonymity.

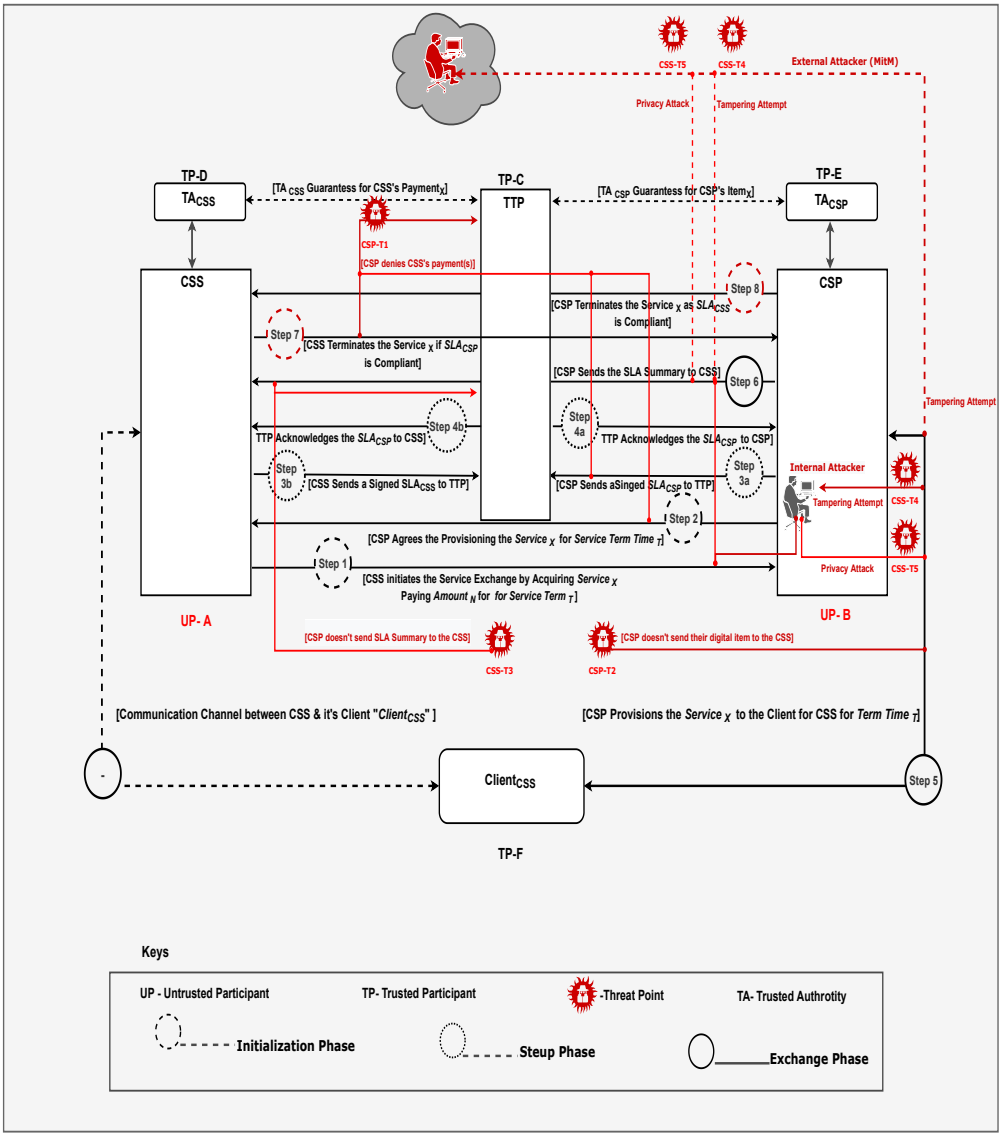


Figure 6.4: CSP Threat Points

CSP-T2: This presents yet another set of threats where a misbehaving CSP could try not to deliver services at all, delivers them partially, or some service state which is not acceptable and expected by their subscriber. Further, defining the adversary extent, let's say that a service is assumed NOT DELIVERED if it has got poor QoS, broken links, or even a service with unsuitability, uncertainty which eventually ends up causing unscheduled outages. The scope of this threat also covers other elements such as if the CSP tries to provision fewer resources reciprocal of what was originally agreed on the SLA. For instance, it provisions less anticipated CPU cycles, memory/ storage allocation, low bandwidth, or anything which compromises prime SLO defined on the SLA and breaches the exchange fairness or atomicity. All these scenarios could deprive the CSS, who has already paid the service provider upfront and now being victimized by not getting their due digital items or services.

CSP-M2: The above-mentioned threats *(T2)* can also be managed using TTP and TA-based assisted architecture. Whatever is classified under poor QoS or SLA compliance issues, can be alerted to the TTP. The TTP will receive the entire data and the metadata along with CSS's escalation for disputed service elements. The TTP runs its verification processes and comes to a conclusion, in this case, where we assume the CSP has demonstrated cheating their subscriber, will be held responsible. TTP will communicate with the TA_{CSP}, who holds an escrow to resolve such disputes. TTP doesn't directly communicate with the CSP to enquire the case as CSS provided factual sheet can be compared what should have been delivered and what service delivery is made. TA_{CSP} will regenerate the item to compensate the CSS with either a financial refund, service credits, or extending the service term to cover their losses, and that show this problem can be resolved.

CSP-T3: We assume that once the SLA is signed, the service exchange is initiated and delivered. The CSP is obliged to send a periodic SLA summary. This informative document is a vital piece of evidence as (a) it convinces and justifies the CSS for their true return of investment *(RoI)*, (b) this is the only information on basis of which the service recipient e.g. the CSS can tell if the SLA has been honored by the CSP. It tells when and how the SLA was compliant or violated. The CSS can only make their claim on these grounds if the SLA summary shows any SLA violations. Here the misbehaving CSP could turn the table by not dispatching the agreed SLA summary to the CSS, at all or not at least in time. There are other possibilities of how a malicious CSP can deprive the CSS in terms of delivering the correct SLA summary, which will be discussed in (T4). Another possibility can also not be ruled out if the SLA summary was sent by the CSP however, it was intercepted by a middle

attacker who either doesn't send it to the CSS or perhaps sends a modified version of it which impersonates that the action is initiated by the CSP.

CSP-M3: Above discussed threats where CSP intentionally fails, it's an obligation by not dispatching the SLA summary can invalidate the service exchange as the CSS won't be able to reconcile the QoS against the SLA hence potentially oppressed economically by the CSP. Incorporating TTP and TAs can also resolve the problem. A CSS, upon not receiving the SLA summary, can initiate the complaint through the TTP to resolve the matter. The TTP who also holds a copy of SLA *(which was supplied by both these participants)* verifies the claim and instructs the CSP to send the SLA summary so the CSS can perform their reconciliation for either a potential service renewal if the service was SLA compliant, if not they can claim their due refund. In case, if it proves that the provisioned service wasn't SLA compliance the TTP will again be approached to arrange a refund by directing the TA_{CSP} to resolve the matter. Implanting a secure coprocessor can also facilitate which will ensure the SLA summary is automatically sent to the CSS on the scheduled date/time. It certainly eliminates any manual processing or an adversarial process infiltration by the CSP to mitigate these threats. Lastly, the middle attacker can also be dealt with by implementing fully homomorphic encryption *(FHE)* which can repel any potential attackers who try to compromise the untrusted communication channels. Even if they even manage to break in or any attempt towards data exfiltration, due to FHE implementation, they won't be able to gain any useful information out.

CSP-T4 A CSP can become an adversary by performing various tampering attempts on both the CSS data and of those digital artifacts they gain while computing or processing their requests. The results are not only produced on their infrastructure but also stored, networked, and of course, transmitted to the CSS or on their behalf as agreed. Some generic tampering may involve as:

- Intentional modification to SLOs such as ART, Latency
- Malicious, unsolicited tampering to system/ infrastructural configs
- Modification/ tampering to results or SLA summary to mimic variant results to constitute an impression of SLA compliance rather than SLA violation.
- Modification/ tampering to Access Control List (ACL) to either downgrade authorized users approved by the CSS or to perform unauthorized privilege escalation of CSS users or CSP's own data center engineers.

- The possibility of an external attacker can also not be ignored as if using untrusted networks or security vulnerabilities they can compromise and tamper the message exchange by leaving no footprints behind by avoiding detection.

CSP-M4: Various elements of the above threat entail one single agenda that how to prevent the tampering whether it's being performed by the CSP or perhaps an external attacker. The ultimate solution to this problem is wrapping up both the processing units and communication channels so both these avenues become (a) inaccessible to both the CSP or the external attacker. The most secure and appropriate mitigation within the CSP's trust boundary is to deploy secure coprocessors such as IBM secure coprocessor family eg IBM4578 or any of their predecessor CPUs. Implementing this secure module, provisions an ultimate security shield through which a malicious CSP or any of the exchange participant cannot tamper or make any similar unsolicited changes to some agreed processes or data, to cheat the other. Similarly, if the data passes through these secure modules, a similar arrangement on the message recipient end can gain some meaningful information out of it. Keeping middle external attackers in mind, we also suggest putting another security shell by implementing fully homomorphic encryption, which would eliminate any remaining threats whatsoever from attackers aiming the data-on-flight through untrusted communication channels.

CSP-T5: CSP can exploit the client/ CSS data privacy. The external attacker can also exploit the same by attacking the mediating communication channel]. When a CSP acts maliciously, to gain some financial or technical gains and perhaps to feed some other curiosity if they can attempt to tamper their customer's data, certainly they are also fully capable of compromising data privacy. This gain ultimately costs the customer then and there or maybe later as being the infrastructure owner a CSP can intercept, record, and sell customer's data. Privacy exploit can also be threatened by external attackers, who can sniff the network traffic between two participants and can benefit the service provider to earn some reputation, fame, or even financial means. If the external attacker can compromise the untrusted channels using various tools, techniques, these sort of privacy compromises can bring the victimized party to their knees.

CSP-M5 Privacy threat also extends CSS challenges to an extra mile. If CSP compromises their subscriber's data privacy and when CSS doesn't have any access to obtain any evidence, it put the CSS in a situation. CSP remains undetected while compromising data privacy which might be too sensitive or

business-critical. The problem can also be resolved by implementing security modules at the processing points and adapting homomorphic encryption guarantees the data at transit will be secure.

6.1.4 Service Exchange Interactions

The following steps show, how participants interact with each other, while a basic service exchange is being performed. This includes interaction setting up, initialization, exchange, abort/ termination phases, and finally, it briefly explains, how dispute resolution work. This interaction would ascertain, the foundations and theory of need, when and why a new architecture is inevitable, and how it would resolve many issues, in the modern cloud era.

Setup Phase

- ***Step 1:*** At the initialization phase, both the main participants establish their service deal when the CSS accepts an intended cloud service or even a compute functionality advertised by the CSP against some specific amount. This service will be processed on CSP-owned infrastructure or data centre facilities. CSS sends an upfront payment token along with service specification (e.g. *compute, network, storage, etc.*) to the CSP. Within Step 1, CSS also defines the service term, SLA (highlighted with certain SLOs) specifications in it so CSP can review and agree upon them.

- ***Step 2:*** CSP, upon receiving the message via Step 1, reviews it against these on-demand services, sought by the CSS. It verifies the service availability, provisioning implications, and term time and payments made by the CSS to validate them all on a mutually agreed SLA. If all items are correct and acceptable, the CSP will proceed, otherwise, will decide to quit the protocol, by letting the TTP know about this termination.

Initialization Phase

- ***Step 3a & Step 3b:*** Once agreed, both the CSP and CSS send signed copies of the SLA and acknowledge their receipts using Step 3a and Step 3b, respectively.

- ***Step 4a & 4b:*** Upon receipt of the signed SLA by both these participants, TTP dispatches its acknowledgment to both, the CSS and the CSP separately, through Step 4a and Step 4b.

Exchange Phase

- ***Step 5:*** This step indicates that CSP provisions the on demand X cloud services at the Client$_{CSS}$ environment on behalf of the CSS at TP-F point as shown on the Fig 6.3*(CSS Threat Points)*. The service will be initiated until the end of the agreed term on the client site *TP-F*.

- ***Step 6:*** CSP is obliged to send a periodic SLA summary to the CSS at the end of each service term for reconciliation purposes. This transmission helps the CSS to ascertain whether there are any potential SLA violations during the service term or if the entire service was delivered as SLA compliant. If the SLA summary is not sent by the CSP, the CSS, being the service subscriber won't able to verify the SLA contents and will be losing their return of investment *(RoI)*. Upon receiving and reviewing these statistics, enable the CSS to either approve the final payment tokens via TA$_{CSS}$, or, instruct, not to proceed if there are any ambiguities are discovered. There is a potential threat where (i) CSS, itself can tamper the SLA summary as depicts under *(CSS-T2)* to fabricate some SLA violations so to avoid payments or even claim falsified service credits/ refunds or (ii)an external attacker *(MitM)* can modify the SLA summary so when it gets to the CSS, it depicts those arbitrary SLA metrics and cause a deception.

Abort/Termination Phase [*Normal/ Abnormal Termination*]

- ***Step 7:*** In an idealistic environment, if both the CSS and the CSP have been honest while exchanging their items, there, normal protocol termination can be expected. CSS will initiate this termination indicating having a fair service delivery, unless they want to have it renewed or in case of the dispute arises, the termination will not occur. Furthermore, if CSS is satisfied with the CSP's service provisioning, till the end of the agreed term, either the CSS terminates the channel, indicating the end of the service. However, there is a flip side of this scenario that when the CSS, decides to act maliciously by committing an exceptional termination because CSS did not want to pay to the CSP showing as a potential threat *(CSS-T1)* for some reason. This will be caught during a future reconciliation done by the CSP, who, in that case, will approach the TTP claiming that the CSS violates the SLA. The dispute resolution procedure is triggered.

- ***Step 8:*** Finally, if all goes smooth, CSP also acknowledges the protocol's normal termination until he receives a service renewal request made by CSS. Now, if there is any dispute between them, the TTP takes care

of those calls by liaising with every participant's corresponding trusted authorities such as TA_{CSP} or TA_{CSS}.

Dispute Resolution Dispute resolution constitutes a state when either party raises an alarm that the other participant has been unfair to them by either not making the payment or not sending the expected item in return. Above mentioned scenario also demonstrates some threat points, during an end-to-end service exchange. For instance, a dispute could arise when the CSS acts maliciously such as *CSS-T1* when they do not pay their dues to the CSP and on anther point *CSS-T2* when they manipulate SLA summary by tampering with service stats faking SLA violations, to avoid any payments. The flip side also shows, when when the CSP demonstrate the similar misbehaviour as illustrated through *CSP-T1*, *CSP-T2*, *CSP-T3*, *CSP-T4* and *CSP-T5* detailed above. This depicts the basic service exchange threat posture of how the CSS can be deprived, who will be obliged to seek some sort of compensation from a trusted third party *(TTP)*. This brings fork a state where the disputes can be dealt with manually or by introducing some novel architecture, the entire process can be automated where there are least chances of having any disputes are arose, from either participant.

1. **Outsourced Dispute Resolution Protocol *(ODRP)* [With Dispute Resolution]** While observing the above mentioned service exchange as Fig 6.1*(Service Exchange Basic Model)*. where both CSS and CSP, do not trust each other, therefore, they act cautiously by outsourcing their liabilities through external guarantors called trusted authorities (TAs). The TTP being a mediator, are duly authorized by the exchange parties by having a mandate to either enquire or direct to regenerate CSS or CSP's items respectively, in case either of them does not behave as expected.

2. **Automated Dispute Resolution Protocol *(ADRP)* [Without Dispute Resolution]** Another dimension to resolve the dispute resolution automatically by ensuring an end to end security, privacy, anonymity more effectively and efficiently to use encryption and anti-tampering modules whether the data is being at rest, transit even going through some compute processes. This is the novelty of our new proposed architecture that the data will eventually be decrypted at some stage, this phenomena raises concerns and associates a lot of supplemented risks especially when data is being decrypted at the service provider's end, kept within their trust boundaries or even while traveling through untrusted channels, the privacy and security risk are intact. To overcome these risks we suggest an architecture based upon fair exchange protocol, using a combination

of encryption and anti-tampering innovations for absolute peace of mind, which will be described shortly. Our architecture is not only offering security at both the participant's end but it also wraps extra security layers to protect the information while the data is traveling through wires.

6.2 Mitigating Malicious Participants

Critically reviewing the basic service exchange model and associated threats potentially expected from either CSS or CSP as highlighted in Fig 6.5 (Union of threat points - CSS & CSP). This mapping introduces some key threats which give an impression that at what stage, which prime service exchange entity could act maliciously to gain some benefits suits them.

Presenting a basic service exchange model draws a correlation of a state when cloud services are being provisioned. Cloud based services are deployed for a cost paid by their subscribers for an agreed service term. Service provisioning comes with some assurances and guarantees which includes ensuring data security, privacy and availability along with a good acceptable QoS. These major service aspects are primarily monitored, managed and supervised by CSPs. There are other subsequent entities can be involved such as TTPs and TAs. Digital services are often provisioned to the end user sites on behalf of CSS. We assume that among these six main service exchange participants, TTP, TA and the Client all are trusted participants and for the time being, discussing their security and misbehaviour is out of the scope for this discussion. Our research demands to focus on the remaining two participants e.g. CSS and CSP and their interaction with each other and how they can misbehave and circumvent the base line security arrangements. Misbehaving participants could also extend their curiosity by compromising properties features a digital service exchange for instance the exchange fairness, atomicity, correctness, autonomy, timing constants, effectiveness, efficiency, trust and non-repudiations.

There are few solutions presented as discussed in Chapter 3 however, this research couldn't find such a solution that can empower the cloud services in terms of upholding the problem of fair exchange aiming an autonomous SLA enforcement solution that covers the majority of the highlighted threats within a single architecture. The previous section highlights various threats and also discusses potential mitigation avenues.

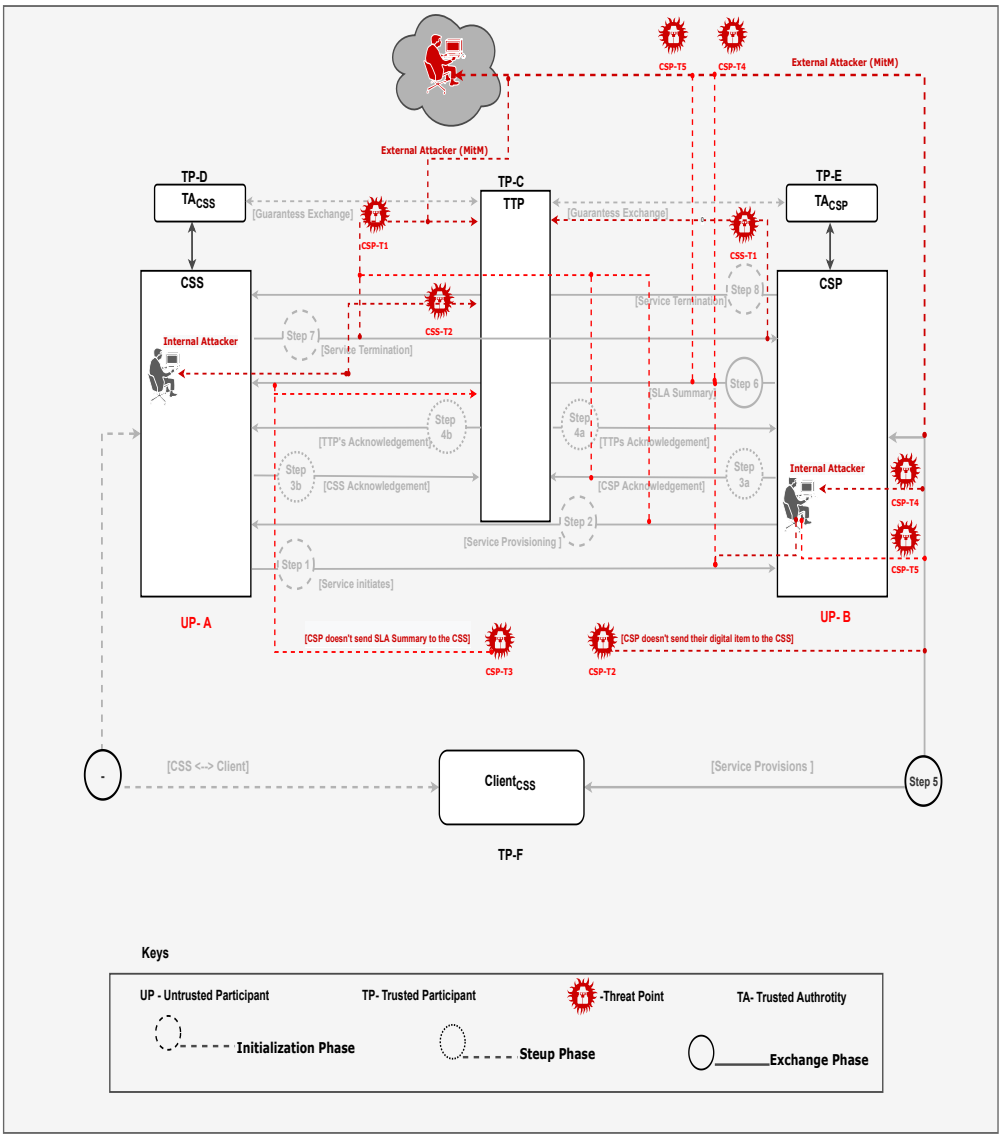


Figure 6.5: Union of Threat Points *(CSS and CSP)*

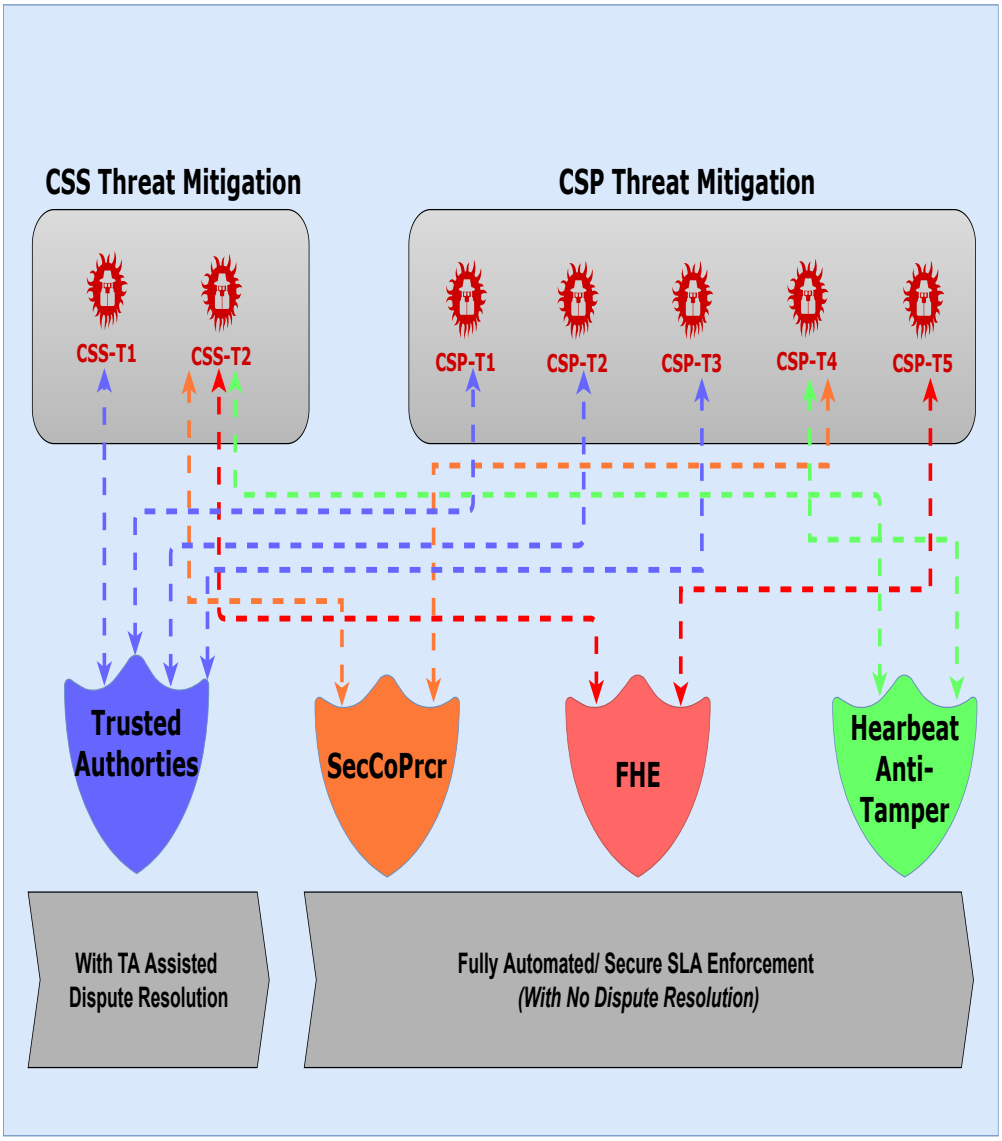


Figure 6.6: Architecture's Threat Mitigation

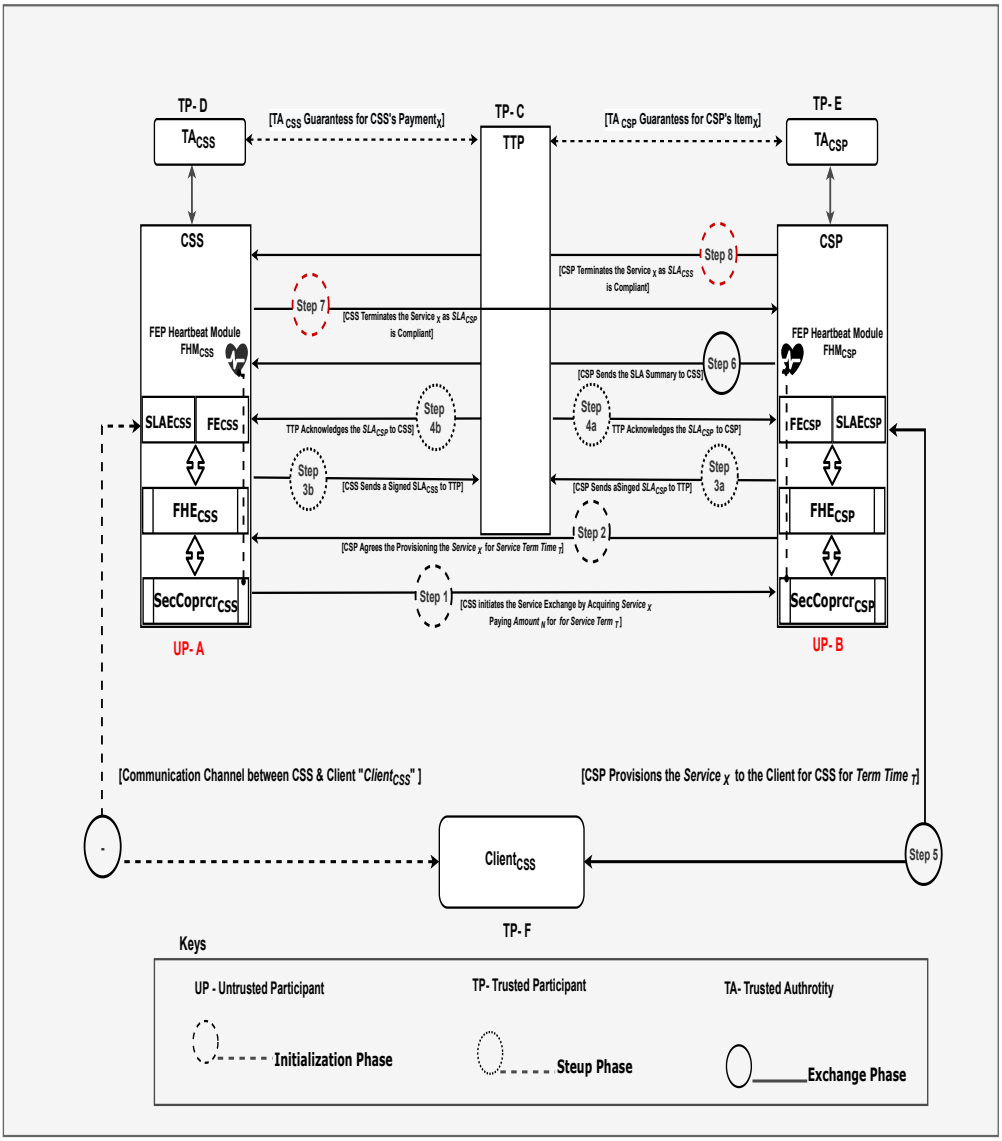


Figure 6.7: Proposed Architecture(II) *(when participants are malicious)*

Chapter 7

Discussion

Every single innovation always brings some safety and security concerns, therefore, manufacturers or technology regulators do publish prevention measures against those potential risks. Since cloud-based service provisioning has become a business norm as all the known industries have somehow commissioned these platforms, similarly, cloud services do bring a variety of risks with them. A CSP owns a slightly bigger chunk of such liabilities and obligations as they own their data centres client's data is processed, stored, shared, and transmitted. Some of the critical risky factors are also secretly shifted towards the service subscribers by these CSPs. Violating good usage policy, service & data infringements, licensing obligations, over usage of virtual resources (by cracking the security controls), non-payments, QoS monitoring, and claiming service credits are some of those issues which could occur during a service provisioning.

A CSP on another side with multi-tenancy capabilities could be even more prone to such misbehavior by stealing systems resources *(CPU cycles, memory or storage)* from one of their low prioritize client and deploying them to a high profile customer for increasing their profit are also possible[96, 101, 194]. Such malicious intentions are potentially be carried out while leaving no tangible forensics evidence so that a less-privileged CSS cannot detect and claim such technical theft. While working on the above-mentioned proposed architectures, we also considered malicious attackers, who could either be working for their unfair economical gains asking for serious ransom money or they might be state-sponsored actors trying to target a business [15, 184, 199]. The work on this thesis was initiated with a conceptual model[160], considering the SLA enforcement is a problem of fair exchange. The causality behind this approach was the service provisioning when CSS signs service and the CSP deploys the service on CSS's platform. In this chapter, readers will be able to examine how multiple security and economic scenarios are treated when a cloud service is deployed. These diverse operating states urge for a solution that alleviates the

prime objective to have automated SLA enforcement in place. Additionally, it also capable of ensuing protective security layers shielding participant's data and privacy with minimal trust deficit throughout the exchange process. The solution can also collaborate occasional and measured assistance of trusted parties such as TTP and TAs.

7.1 Protocol's Varying Scenarios & Challenges

We initially marked for two service exchange scenarios e.g. *(a)* when our nominated participants *(CSP and CSS)* are either *loss averse* OR (b) when either or both participants *act maliciously* so to have unsolicited gains over the other participants which is the prime objective of FEP. Here both the analysis can be found, which discusses architecture implementations, operating environment, assumptions, challenges, and scope. Following the discussion will not only cover the CSP or the CSS but it would also frame how external attackers can attempt to break the running cloud services or perhaps target communication mediums on untrusted networks where they can steal some useful information or at least they can disrupt business as usual (BAU) services or cause a process failure or modify system behaviour.

7.1.1 Architecture I: Implantation Scenario, Challenges & Scope

While working on the prototype implementation, several avenues, challenges, and properties were studied to achieve results for our FEP based architectures. Following are some of those bullet points which give insights into our implementation and shed light on different research dimensions and other avenues.

Possible TTP Implementation: Typically, a TTP in fair exchange is thought of as a trusted "institution", which will be able to enforce the fair exchange. However, this does not have to be the case and, in certain cases such as SLA enforcement, not desirable. On the realization of TTP, my research envisages that it can be implemented as a replicated state machine using the atomic broadcast protocol of Internet-scale systems like ZooKeeper [98]. However, if TTP nodes can experience Byzantine faults (malicious intrusions), then robust protocols such as [45] need to be used; these protocols involve several rounds of message exchanges and may not scale. Alternatively, one could also explore the emerging Blockchain technology combined with smart contracts. Our colleague's recent work [128] on eVoting using Ethereum Blockchain motivates the possibility of similarly building TTP for scalable fair-exchange applications.

Fair Exchange Issues: My SLA enforcement protocol requires the TTP to be active, i.e., the protocol requires an online TTP, increasing the communication overhead. However, this needs not to be the case since we assume the parties to be loss averse, hence an optimistic fair exchange type of protocol would have sufficed. However, in line with our future work on making the SLA enforcement resilient, we have chosen not to pursue an optimistic FE protocol. Similarly, given loss averse parties, nonces are not strictly required as loss averse participants rule out replay attacks. Finally, our protocol presents stateless TTP, i.e., the state is reset once a given round is complete. However, the protocol can be easily instrumented to make state persist across multiple rounds.

Architecture Limitations: As mentioned previously we collected these results on separate cloud-based entities rather than monitoring them on the end-user‘s estate. A CSP can only guarantee a request-response when it leaves its trusted boundary into the public net. How long would it takes to get to the end user‘s application layer comprises a multitude of factors, of which firstly most of the CSP‘s don‘t take the responsibility at all, if some of them do, they would add up the cost for premier service delivery for this purpose

7.1.2 Architecture II: Implantation Scenario, Challenges & Scope

This subsection debates our second approach, where we extended our first architecture by *(a)* swapping participants states from loss aversion to a state where participants are malicious *(b)*we change the TTP deployment from an in-line to on-line state and *(c)* finally we deploy trusted authorities, sub-contracted entities, on behalf of both, the CSP and CSS. CA play a vibrant role for verification and validation purposes, while cloud service exchange is being performed and how their feedback to the TTP disseminates on-demand item guarantees.

Shedding some light on the concept, when a trading participant can act maliciously, is the most key topic within the modern-day electronic trading realm. Trust deficit brings the theory of escrow agreements and services. Internet-related fraud is not a new phenomenon. Reports suggest that global e-commerce based fraudulent figures might touch $25bn by 2024 [144]. Businesses, therefore, are quite conscious when making deals based on e-commerce because of this trust deficit [90]. Cloud service falls into the same category where various global cloud services showcase their digital services and market them using very attractive features however when the actual service is deployed to the client site, the experience gets bitter. CSPs don’t keep their promises and after

the service deployment cooling off period, their QoS either gets compromised or sometimes they intentionally misbehave by not maintaining the service stability, availability, or unauthorized cloud resource allocation to other service tenants, which violates the SLA [190]. The entire transaction is meant to be recorded and appropriate timely mitigation is promised however, cloud services somehow manage to cheat their service subscribers. The behavioural detection of a cheating business entity holds a lot of complexities and challenges within. Stealing cloud resources [95, 172, 193] and making other potential cheating attempts are known to the media when CSPs commit such cheating [65]. Sometimes external actors [38] are also behind some service disruptions to cause service outages [55, 170]. Some service subscribers could also act maliciously when they refuse to make their payments or they perform some service infringements [19, 44].

Possible TTP Implementation: In this architecture, the same assumption was carried out for the TTP for them being a trusted entity same as corresponding trusted authorities for CSS and CSP. In terms of TTP placement, which makes a lot of difference in terms of calculating their response time while entities are interacting with each other. Keeping the TTP's online constitutes these factors as its involvement would be only considered when necessary such as handling disputes. While testing the architecture(I), was presented in Chapter 5 implementation was carried out by interconnecting architectural nodes by configuring its prime components on diverse cloud environments, however, in the case of this architecture which holds a broader scope when mitigating diverse set of *internal and external* threats.

TTP correlates on multiple communications channels while reviewing potential disputes, however, the novelty of this design is that it requires least minimal interaction to conduct thorough process and SLA verifications. Our architecture, equipped with multiple security layers which makes it confidently secure. TTP does review security feeds from the core component *e.g.* secure co-processor modules *(SCM)*, which sits at the very heart of this architecture. It seamlessly protects data aiming at both security and privacy. A concern question, that *who will be watching the watchers* is also addressed and promptly mitigated by adding the module, monitoring the SCM potential tampering. In case of a malicious actor, gets hold of the SCM firmware code possibly from the dark web so to attack this module, the additional security aperture would protect it. It smartly implements anti-tampering signals using a module based upon heartbeat protocol [56, 100]. This module is capable of beaconing real-time alarms if it detects any tampering attempts that may have been performed by one of the participants. Such additional security arrangement is deployed on both CSS and CSP estate to ensure a degree of fairness as

well as data security. Finally, this architecture also considers data *@ transit)* might intercepted by some *(M.i.T.M)*, who can rage attacks on untrusted communication channels, therefore, the implementation puts full homomorphic encryption *FHE* which nails out this issue too.

This chapter cross-examines both the proposed architectures. It evaluates and discusses how and what the protocol's deployment can achieve when the participants are believed to be loss averse and secondly when they are planning to misbehave. Security constraints, resilience, and futuristic approach were the focal points therefore, technologies like trusted modules, end-to-end data encryption, and module's security were ensured. We discussed the role of TTP and associated TAs along with the potential possibilities of external attack vectors. The next and the final chapter gives a conclusion and leaves some thoughts for future work and extended possibilities where easing the assumptions, switching the protocol from a synchronous to an asynchronous model, and adding some variations about TTP's role would be quite interesting factors.

Chapter 8

Conclusion and Future Work

This research studied the cloud service fabric, focusing on SLA enforcement regarding participants' potential misbehavior. SLA being the key instrument is framed by numerous tools, techniques, and procedures for compliance purposes. Both the open-source and propitiatory solutions and frameworks, predominantly focus either on SLA monitoring or detection through devising centralized configuration management tools and related APIs. This constitutes the cloud service provider mandate, which leaves elusive and limited options to allow the service subscriber to share true QoS stats mirroring the low-level service metrics while the service is being provisioned. Our proposed design of a potential framework certainly opens another avenue by introducing a significant shift of cloud service control and its provisioning, for both the CSP and the CSS, by enforcing the SLA's through a strictly impartial and fair-centric fashion.

Defining, implementing, and enforcing security balanced cloud SLA plays a pivotal role for both CSS or CSS during cloud service provisioning. SLA enforcement holds the strategic fairness aspect from initial service deployment till the end of the service term. Accomplishing cloud service objectives against the actual delivered service and its capabilities, functionalities, critical dependencies can only be evaluated when SLA enforcement can be ensured. This chapter primarily concentrates on the fact that although current cloud SLA monitoring and detection arrangements are there however such measures are inadequate to deal with today's threat landscape and other well-known regulatory and legal requirements. The end of the service term often leaves one or perhaps both the CSP or CSS, with some vague stats and service reconciliation, which would not justify a true return of investment *(RoI)* until individual and targeted service elements are examined using low-level infrastructure investigations using approved digital forensics techniques to evaluate service metrics are SLA compliant or not? This is generally not feasible and mostly not permissible by the corresponding CSP, especially when cloud-based

services are further sub-contracted using various outsourced global brokerage service channels. This research tackles such SLA enforcement instability by classifying it as a fair exchange problem. Fair exchange protocol is perhaps the closest as well as the logical solution, which sets a well-defined equilibrium to facilitate the SLA enforcement.

While subcontracting cloud services and signing their respective SLAs, ignoring end-to-end security and privacy protection elements of an entity's data *(at transit or rest)*, could become a serious risk. If one of the participants decides to misbehave or even an invisible malicious middle actor, tries to sabotage service provisioning by penetrating insufficient security layers, targeting any segment of a cloud service provisioning, can leave serious business damages.

Our proposed architectures in this thesis, fair exchange protocols, and their sub-protocols are embedded using such a novelty, which not only eliminates multiple technical deceptions possibilities but also adds additional security layers to protect from various threats, posed by unknown malicious attackers during a cloud service provisioning. These attempts, if are successful could easily aim to deprive the targeted participant, while the service exchange is being carried out. Our framework design also enforces other strategic security features such as privacy and data security, shielding potential tampering attempts. These attempts can easily disrupt an operational level or technical level workflow of a web service. It can modify expected service behaviour through mutilating resource availability, response time, latency, the ratio of fault detection, fault tolerance, and resolution. Other issues such as unsolicited service outages could compromise the persecuted partner's business repute as well their technology investment and production line.

In contrast to previous approaches, this research benefits by filling potential gaps by integrating trusted modules such as Secure Coprocessors (addresses data at rest), the Fully Homomorphic Encryption (addresses data at transit), and other security modules to add an additional protective layer which makes it even harder for an attacker or a malicious participant to attempt a service stats, configuration changes or any kind of data modifications. As within the information security industry, the fact is well known and well understood that there is no 100%security, therefore, there will be some limitations that could still be exploitable by the shrewd attackers, who could still make holes to these security walls and might tamper data at transit using modern untrusted communication channels using special offensive tools and techniques to break the TMP's firmware. As the problem of SLA enforcement is well understood, ensuring the QoS and the QoE are only achievable, if the SLA enforcement is

automated and correctly configured on each participating node. It will eventually facilitate fairly distributed services and their corresponding elements. Our proposed architecture restricts SLA obligations to each participant by enforcing uninterruptible and trusted SLA monitoring, detection, and enforcement in a single framework. It measures service values for each distributed segment and produces transparent cloud service reconciliation metrics for a forensic-oriented investigation, for any unresolved service disputes if arise.

Previous works resolve other associated problems partially were substantial challenges still intact hence do warrant a solution to alleviate them. Research studies various proposed methodologies focus on all the service element issues from low-level metrics to high-level service availability in such an automated way where the challenging SLA enforcement would be difficult and does not present any sort of operational ambiguity, once tested in the real cloud environments. A methodology should also consider data privacy and security elements where detecting and resolving SLA violations could ensure defense-in-depth.

As autonomous service control systems are rapidly taking over the digital industry. Cloud service deployment is also taking a huge technological shift from manually observed cloud services such as QoS controls are transitioned towards autonomous monitoring and control sensors where the least human interaction is needed. Although our suggested framework is in its early evolutionary phases to combat the big and mystified cloud SLA problem, however, it stills demonstrates its capabilities towards SLA enforcement with extended security layers and smart controls. In the presence of multi-parties using fair exchange, future objectives are to further analyze other potential assumptions by flipping timing models, and testing other deception and attacking vectors and techniques to break the security arrangements and infringe SLA enforcement. Fault models can disclose novel and harsh challenges, which might empower the service providers to claim unprecedented service controls to impact the overall fairness. A participant with dishonest intent could take over by committing some infringement techniques affecting protocol and the sub-protocol modules. Extended studies warrant observing what technical, operational, regulatory, and legal requirements can be neutralized where some of the protocol's properties especially non-repudiation, timeliness, and fairness can be sabotaged. SLA enforcement holds immense research opportunists to improve and extend various use cases which can constitute cloud service assurance level for every stakeholder at the end of the term. Architecture's deployment to multiple cloud environments might discover new constraints related to the TTPs and CAs fault models such as Byzantine failures.

Our overall concept mainly spins around two scenarios, loss-averse participants and malicious participants who interact with each other through outsourced entities such as TTP and the TAs using a synchronous communication model. Future work would also examine, how to attain more fair exchange properties, by switching these service arrangements through asynchronous communication model and quantifying threat modeling. This would include help achieving SLA enforcement using cross-platform cloud service functionalities and their QoS dependencies in a more resilient through extended module configuration. Real-time service negotiations and dispute resolution in a multi-tenancy cloud atmosphere can enhance the investigative avenues. There is an honest intention to collaborate diverse service deployments using other frameworks to intuit our architecture's constraints. Extending our research towards SLA enforcement would also anticipate some natural considerations which could suggest protocol's future optimization empowering fairness, privacy, security, autonomy, and trustworthiness. Our protocol can also be blended by SOA frameworks such as IT Service Management *(ITSM)* [148], or similar standards empowering assurance.

It would also be extremely useful and inevitable at a later stage to redefine our proposed architecture to comply and integrate modern security frameworks such as COBIT [103], NIST [61, 86] Security Orchestration, Automation, and Response *(SOAR)* [104], Cloud Controls Matrix [57] best practices along with Blockchain, and AI-centric platforms, as our future extension. Such provisions will certainly augment SME's capabilities to have a better control attaining and sharing reliable SLA metrics for their forensics soundness and can easily be integrated into well-known forensics frameworks as well for investigating potential cloud-based unresolved disputes if participants do not oblige such automated arbitrations.

Bibliography

[1] Electronic signatures in global and national commerce act. *PUBLIC LAW 106–229—JUNE 30, 2000*. URL `https://www.gpo.gov/fdsys/pkg/PLAW-106publ229/pdf/PLAW-106publ229.pdf`.

[2] ACPO. Acpo good practice guide for digital evidence march 2012. #mar# 2012. URL `http://library.college.police.uk/docs/acpo/digital-evidence-2012.pdf`.

[3] Syed Zubair Ahmad and Mahmood Ashraf. A seamless qos provisioned micro-mobility management scheme. In *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*, pages 480–485. IEEE, 2004.

[4] Mustapha Ait-Idir, El Hadi Cherkaoui, Elie Rachkidi, Nada Chendeb, and Nazim Agoulmine. Most-cb: Sla enforcement and smart vne (virtual network embedding) in a multi cloud providers environment. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 86–92. IEEE, 2014.

[5] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. Elasticity in cloud computing: State of the art and research challenges. *IEEE Transactions on Services Computing*, 11:430–447, 2018.

[6] Rizik MH Al-Sayyed, Hijawi Wadi'A, Anwar M Bashiti, Ibrahim AlJarah, Nadim Obeid, and Omar YA Al-Adwan. An investigation of microsoft azure and amazon web services from users' perspectives. *International Journal of Emerging Technologies in Learning (iJET)*, 14(10):217–241, 2019.

[7] Abdullah M Alaraj. Simple and efficient contract signing protocol. *arXiv preprint arXiv:1204.1646*, 2012. URL `https://arxiv.org/pdf/1204.1646`.

[8] Mohammed Alhamad, Tharam Dillon, Chen Wu, and Elizabeth Chang. Response time for cloud computing providers. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, pages 603–606. ACM, 2010.

[9] Esraa Alomari, Selvakumar Manickam, BB Gupta, Shankar Karuppayah, and Rafeef Alfaris. Botnet-based distributed denial of service (ddos) attacks on web servers: classification and art. *arXiv preprint arXiv:1208.0403*, 2012.

[10] Abdullah AlOtaibi and Hamza Aldabbas. A review of fair exchange protocols. *International Journal of Computer Networks & Communications*, 4(4):307, 2012. URL `http://airccse.org/journal/cnc/0712cnc20.pdf`.

[11] Amazon. Amazon s3 service level agreement, 2015. URL `https://aws.amazon.com/s3/sla/`. Last Updated September 16, 2015.

[12] Amazon. Summary of the amazon s3 service disruption in the northern virginia (us-east-1) region, #mar# 2017. URL `https://aws.amazon.com/message/41926/`.

[13] Inc. Amazon Web Services. Aws sdk for java developer guide. Electronically, #aug# 2018. URL `https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/java-dg-exceptions.html`. AWS Documentation » AWS SDK for Java » Developer Guide » Using the AWS SDK for Java » Exception Handling.

[14] AmazonAWS. Aws elastic beanstalk easy to begin, impossible to outgrow. Electronic, #jun# 2018. URL `https://aws.amazon.com/elasticbeanstalk/`.

[15] Oxford Analytica. Higher pay-offs will fuel ransomware. *Emerald Expert Briefings*, (oxan-es), 2019.

[16] S Anithakumari and K Chandrasekaran. Negotiation and monitoring of service level agreements in cloud computing services. In *Proceedings of the International Conference on Data Engineering and Communication Technology*, pages 651–659. Springer, 2017.

[17] S Anithakumari and K Chandra Sekaran. Autonomic sla management in cloud computing services. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 151–159. Springer, 2014.

[18] Apache. Apache jmeter introduction, #jun# 2018. URL `https://jmeter.apache.org/`.

[19] N Asokan, Els Van Herreweghen, and Michael Steiner. *Towards a framework for handling disputes in payment systems*. IBM Thomas J. Watson Research Division, 1998. URL

http://static.usenix.org/legacy/publications/library/proceedings/ec98/full_papers/asokan/asokan.pdf.

[20] Nadarajah Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 7–17. ACM, 1997.

[21] Nadarajah Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 7–17. ACM, 1997.

[22] Nadarajah Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, pages 86–99. IEEE, 1998.

[23] Nadarajah Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 591–606. Springer, 1998.

[24] Patrick S Atiyah. Contract and fair exchange. *The University of Toronto Law Journal*, 35(1):1–24.

[25] Gildas Avoine, Felix C. Freiling, Rachid Guerraoui, and Marko Vukolic. Gracefully degrading fair exchange with security modules. In *EDCC*, 2005.

[26] Alireza Bahreman and JD Tygar. Certified electronic mail. Master's thesis, Citeseer, 1992. URL http://people.ischool.berkeley.edu/~tygar/papers/Certified_Electronic_Mail/Certified_electronic_mail.pdf.

[27] Charu Bamboriya and SR Yadav. A survey of different contract signing protocols, 2014.

[28] Feng Bao, Robert H Deng, and Wenbo Mao. Efficient and practical fair exchange protocols with off-line ttp. In *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, pages 77–85. IEEE, 1998. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=674825.

[29] Salman A Baset. Cloud slas: present and future. *ACM SIGOPS Operating Systems Review*, 46(2):57–66, 2012.

[30] Salman A Baset. Cloud slas: present and future. *ACM SIGOPS Operating Systems Review*, 46(2):57–66, 2012.

[31] Monica Belcourt. Outsourcing—the benefits and the risks. *Human resource management review*, 16(2):269–279, 2006.

[32] Michael Ben-Or, Oded Goldreich, Silvio Micali, and Ronald L Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.

[33] Richard Berger, Rene Peralta, and Tom Tedrick. A provably secure oblivious transfer protocol. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 379–386. Springer, 1984.

[34] Tim Berners-Lee, Roy Fielding, and Henrik Frystyk. Hypertext transfer protocol–http/1.0. Technical report, 1996.

[35] Elisa Bertino, Lorenzo Martino, Federica Paci, and Anna Squicciarini. *Security for web services and service-oriented architectures*. Springer Science & Business Media, 2009.

[36] Srimoyee Bhattacherjee, Sunirmal Khatua, and Sarbani Roy. A review on energy efficient resource management strategies for cloud. In *Advanced Computing and Systems for Security*, pages 3–15. Springer, 2017.

[37] GR Blakley and I Borosh. Rivest-shamir-adleman public key cryptosystems do not always conceal messages. *Computers & mathematics with applications*, 5(3):169–178, 1979.

[38] David Bond. Hackers target cloud services, attackers spread malware through it service providers. Financial Times, July 2018. URL `https://www.ft.com/content/4f990a78-537a-11e8-84f4-43d65af59d43`. Security and Defence.

[39] Sumit Bose, Anjaneyulu Pasala, Dheepak Ramanujam, Sridhar Murthy, Ganesan Malaiyandisamy, et al. Sla management in cloud computing: A service provider's perspective. *Cloud Computing: Principles and paradigms, Rajkumar Buyya, James Broberg, and Andrzej Goscinski, Eds. New Jersey, USA: John Wiley & Sons*, pages 413–436, 2011.

[40] Athman Bouguettaya, Quan Z Sheng, and Florian Daniel. *Web services foundations*. Springer, 2014.

[41] Colin Boyd and Ernest Foo. Off-line fair payment protocols using convertible signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 271–285. Springer, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.4650&rep=rep1&type=pdf`.

[42] Ivona Brandic, Vincent C Emeakaroha, Marco AS Netto, and César AF De Rose. Application-level monitoring and sla violation detection for multi-tenant cloud services. *Emerging Research in Cloud Distributed Computing Systems*, page 157, 2015.

[43] Ernest F Brickell, David Chaum, Ivan B Damgård, and Jeroen van de Graaf. Gradual and verifiable release of a secret. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 156–166. Springer, 1987.

[44] Patrícia Raquel dos Santos Cadete. *Marketing plan: FraudProof: a SaaS anti-fraud solution for the digital advertising industry.* PhD thesis, 2018.

[45] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186, 1999.

[46] Amadeo Charlé. Kirti: Decentralized reputation and sla enforcement for cybersecurity. 2020.

[47] Congwu Chen, Lei Li, and Jun Wei. AOP based trustable SLA compliance monitoring for web services. In *QSIC*, pages 225–230. IEEE Computer Society, 2007.

[48] Li Chunlin, Zhou Min, and Luo Youlong. Elastic resource provisioning in hybrid mobile cloud for computationally intensive mobile applications. *The Journal of Supercomputing*, pages 1–32, 2017.

[49] Kassidy P. Clark, Martijn Warnier, Frances M. T. Brazier, and Thomas B. Quillinan. Secure monitoring of service level agreements. *2010 International Conference on Availability, Reliability and Security*, pages 454–461, 2010.

[50] CNSight. Top 5 cyber attacks in the aviation industrytop 5 cyber attacks in the aviation industry. *article*, April 2021. URL `https://cnsight.io/2021/04/16/top-5-cyber-attacks-in-the-aviation-industry/`.

[51] Luigi Coppolino, Salvatore D'Antonio, Giovanni Mazzeo, Gaetano Papale, Luigi Sgaglione, and Ferdinando Campanile. An approach for securing critical applications in untrusted clouds. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 436–440. IEEE, 2018.

[52] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design.* pearson education, 2005.

[53] Benjamin Cox, J Doug Tygar, and Marvin Sirbu. Netbill security and transaction protocol. In *USENIX Workshop on Electronic Commerce*, volume 1, 1995.

[54] Flaviu Cristian. Synchronous and asynchronous group communication (long version). *Personal Communication, avril*, 1996. URL `https://pdfs.semanticscholar.org/efc1/ff91bc301c3d6344cb308b4f619914a0e871.pdf`.

[55] CRN. The 10 biggest cloud outages of 2016, #dec# 2016. URL `http://crn.com/slide-shows/cloud/300083247/the-10-biggest-cloud-outages-of-2016.htm/pgno/0/1`.

[56] Patrick Cronin, Chengmo Yang, Dongqin Zhou, Keni Qiu, Xin Shi, and Yongpan Liu. 'the danger of sleeping', an exploration of security in non-volatile processors. In *2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 121–126. IEEE, 2017.

[57] CSA. The csa cloud controls matrix (ccm) is a cybersecurity control framework for cloud computing. March 2021. URL `https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4/`.

[58] CSCC. Public cloud service agreements: What to expect and what to negotiate version 2.0.1, #aug# 2016. URL `http://cloud-council.org/deliverables/`.

[59] Mohammad Torabi Dashti. Optimistic fair exchange using trusted devices. In *Stabilization, Safety, and Security of Distributed Systems, 11th International Symposium, SSS 2009, Lyon, France, November 3-6, 2009. Proceedings*, pages 711–725, 2009. doi: 10.1007/978-3-642-05118-0_49. URL `https://doi.org/10.1007/978-3-642-05118-0_49`.

[60] Frank S de Boer, Elena Giachino, Stijn de Gouw, Reiner Hähnle, Einar Broch Johnsen, Cosimo Laneve, Ka I Pun, and Gianluigi Zavattaro. Analysis of sla compliance in the cloud–an automated, model-based approach. *arXiv preprint arXiv:1908.10040*, 2019.

[61] Frederic de Vaulx. Nist cloud service metrics model, #jun# 2014. URL `https://www.nitrd.gov/nitrdgroups/images/6/67/De-Vaulx-Cloud-Service-Metrics-Model-20140806-v2.pdf`. Constructing High-Quality Cloud Service Level Agreement (SLAs).

[62] Frederic J de Vaulx, Eric D Simmon, and Robert B Bohn. Cloud computing service metrics description. Technical report, 2018.

[63] Robert H Deng, Li Gong, Aurel A Lazar, and Weiguo Wang. Practical protocols for certified electronic mail. *Journal of network and systems management*, 4(3):279–297, 1996. URL `https://www.researchgate.net/profile/Aurel_Lazar/publication/2823287_Practical_Protocols_For_Certified_Electronic_Mail/links/54372c270cf2bf1f1f2d485d.pdf`.

[64] Alex Depoutovitch, Chong Chen, J. Chen, P. Larson, Shu Lin, Jack Ng, Wenlin Cui, Q. Liu, W. Huang, Y. Xiao, and Yongjun He. Taurus database: How to be fast, available, and frugal in the cloud. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.

[65] Roberto Di Pietro, Flavio Lombardi, Fabio Martinelli, and Daniele Sgandurra. Anticheetah: Trustworthy computing in an outsourced (cheating) environment. *Future Generation Computer Systems*, 48:28–38, 2015.

[66] Dimensiondata. Comparing public cloud service level agreemtns, February 2013.

[67] Yevgeniy Dodis and Leonid Reyzin. Breaking and repairing optimistic fair exchange from podc 2003. In *Proceedings of the 3rd ACM workshop on Digital rights management*, pages 47–54, 2003.

[68] Shewangu Dzomira. Electronic fraud (cyber fraud) risk in the banking industry, zimbabwe. *Risk Governance and Control: Financial Markets and Institutions*, 4(2):16–26, 2014.

[69] Ahmed El-Yahyaoui and Mohamed Dafir ECH-CHERIF EL KETTANI. A verifiable fully homomorphic encryption scheme for cloud computing security. *Technologies*, 7(1):21, 2019.

[70] Vincent C. Emeakaroha, Tiago C. Ferreto, Marco Aurélio Stelmar Netto, Ivona Brandic, and César A. F. De Rose. Casvid: Application level monitoring for SLA violation detection in clouds. In *COMPSAC*, pages 499–508. IEEE Computer Society, 2012.

[71] Vincent C Emeakaroha, Ivona Brandic, Michael Maurer, and Schahram Dustdar. Cloud resource provisioning and sla enforcement via lom2his framework. *Concurrency and Computation: Practice and Experience*, 25 (10):1462–1481, 2013.

[72] Clifton A Ericson and Clifton Ll. Fault tree analysis. In *System Safety Conference, Orlando, Florida*, volume 1, pages 1–9, 1999.

[73] Thomas Erl, Anish Karmarkar, Priscilla Walmsley, Hugo Haas, L Ümit Yalçinalp, Kevin Liu, David Umit Orchard, Andre Tost, and James Pasley. *Web service contract design and versioning for SOA*. Pearson Education, 2008.

[74] SPECS EU. Secure provisioning of cloud services based on sla management, 2016.

[75] Paul Ezhilchelvan and Isi Mitrani. Optimal provisioning of servers for hosting services of multiple types. *Simulation Modelling Practice and Theory*, 75:17 – 28, 2017. doi: dx.doi.org/10.1016/j.simpat.2017.03.011.

[76] Paul D Ezhilchelvan and Santosh K Shrivastava. A family of trusted third party based fair-exchange protocols. *IEEE Transactions on dependable and secure computing*, 2(4):273–286, 2005.

[77] Asma Al Falasi and Mohamed Adel Serhani. A framework for sla-based cloud services verification and composition. In *IIT 2011*, 2011.

[78] Milan Fort, Felix Freiling, Lucia Draque Penso, Zinaida Benenson, and Dogan Kesdogan. Trustedpals: Secure multiparty computation implemented with smart cards. In *European Symposium on Research in Computer Security*, pages 34–48. Springer, 2006.

[79] Matthew K Franklin and Michael K Reiter. Fair exchange with a semi-trusted third party. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 1–5. ACM, 1997. URL `http://www.cs.unc.edu/~reiter/papers/1997/CCS1.pdf`.

[80] Felix C. Freiling, Maurice Herlihy, and Lucia Draque Penso. Optimal randomized fair exchange with secret shared coins. In *Principles of Distributed Systems, 9th International Conference, OPODIS 2005, Pisa, Italy, December 12-14, 2005, Revised Selected Papers*, pages 61–72, 2005. doi: 10.1007/11795490_7. URL `https://doi.org/10.1007/11795490_7`.

[81] Felix C Gartner, Henning Pagnia, and Holger Vogt. Approaching a formal definition of fairness in electronic commerce. In *Reliable Distributed Systems, 1999. Proceedings of the 18th IEEE Symposium on*, pages 354–359. IEEE, 1999. URL `https://pdfs.semanticscholar.org/f8eb/08c721ad5cdfa1cef0bebff20a393d03bfc5.pdf`.

[82] Mahdi Fahmideh Gholami, Farhad Daneshgar, Ghassan Beydoun, and Fethi Rabhi. Challenges in migrating legacy software systems to the cloud—an empirical study. *Information Systems*, 67:100–113, 2017.

[83] Google. Google compute engine service level agreement (sla). Technical report, #nov# 2016. URL https://cloud.google.com/compute/sla.

[84] Rüdiger Grimm and Kambiz Zangeneh. Cybermoney in the internet: An overview over new payment systems in the internet. In *Communications and Multimedia Security II*, pages 183–195. Springer, 1996.

[85] G Grispos, WB Glisson, and T Storer. Calm before the storm: The emerging challenges of cloud computing in digital forensics (august 2011), August 2011.

[86] NIST Cloud Computing Security Working Group et al. Nist cloud computing security reference architecture. *Working document, NIST*, 2013.

[87] GuidingMetrics. The cloud service industry's 10 most critical metrics. January 2018. URL http://guidingmetrics.com/content/cloud-services-industrys-10-most-critical-metrics.

[88] Haryadi S Gunawi, Mingzhe Hao, Riza O Suminto, Agung Laksono, Anang D Satria, Jeffry Adityatama, and Kurnia J Eliazar. Why does the cloud stop computing? lessons from hundreds of service outages. In *SoCC*, pages 1–16, 2016. URL https://pdfs.semanticscholar.org/4f3d/70d5483b461f05c7f92f187cd1913a2434ea.pdf.

[89] Haryadi S Gunawi, Mingzhe Hao, Riza O Suminto, Agung Laksono, Anang D Satria, Jeffry Adityatama, and Kurnia J Eliazar. Why does the cloud stop computing? lessons from hundreds of service outages. In *SoCC*, pages 1–16, 2016.

[90] Yue Guo, Yongchuan Bao, Barnes J Stuart, and Khuong Le-Nguyen. To sell or not to sell: Exploring sellers' trust and risk of chargeback fraud in cross-border electronic commerce. *Information Systems Journal*, 28(2): 359–383, 2018.

[91] Ahsan Habib, Mohamed Hefeeda, and Bharat K Bhargava. Detecting service violations and dos attacks. In *NDSS*, 2003.

[92] Vassos Hadzilacos and Sam Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical report, Cornell University, 1994.

[93] Shon Harris. *CISSP all-in-one exam guide*. McGraw-Hill, Inc., 2010.

[94] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. Data in, fact out: automated monitoring

of facts by factwatcher. *Proceedings of the VLDB Endowment*, 7(13): 1557–1560, 2014.

[95] Ting He, Shiyao Chen, Hyoil Kim, Lang Tong, and Kang-Won Lee. Scheduling parallel tasks onto opportunistically available cloud resources. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 180–187. IEEE, 2012.

[96] Haiyang Hu, Zhongjin Li, and Hua Hu. An anti-cheating bidding approach for resource allocation in cloud computing environments. *J Comput Inf Syst*, 8(4):1641–1654, 2012.

[97] Jiankun Hu, Hemanshu R Pota, and Song Guo. Taxonomy of attacks for agent-based smart grids. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1886–1895, 2013.

[98] Patrick Hunt, Mahadev Kumar, Flavio Junqueira, and Benjamin Reed. Zookeeper: Wait free coordination for internet scale systems. In *USENIX annual technical conference*, page 9, 2010.

[99] Emir Husni, Bramanto Leksono, and Muhammad Ridho Rosa. Digital signature for contract signing in service commerce. In *Technology, Informatics, Management, Engineering & Environment (TIME-E), 2015 International Conference on*, pages 111–116. IEEE, 2015. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7389757`.

[100] Ahmad Ibrahim, Ahmad-Reza Sadeghi, Gene Tsudik, and Shaza Zeitouni. Darpa: Device attestation resilient to physical attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 171–182, 2016.

[101] Joseph Idziorek, Mark Tannian, and Doug Jacobson. Detecting fraudulent use of cloud resources. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 61–72, 2011.

[102] HACKER INTELLIGENCE INITIATIVE. Man in the cloud (mitc) attacks. Technical report, Imperva. URL `https://www.imperva.com/docs/HII_Man_In_The_Cloud_Attacks.pdf`.

[103] ISACA. Cobit. April 2019. URL `https://www.isaca.org/resources/cobit`. COBIT® 2019 is the most recent evolution of ISACA's globally recognized and utilized COBIT framework.

[104] Chadni Islam, Muhammad Ali Babar, and Surya Nepal. Architecture-centric support for integrating security tools in a security orchestration

platform. In *European Conference on Software Architecture*, pages 165–181. Springer, 2020.

[105] ITR. Internet traffic report, #jun# 2018. URL `http://www.internettrafficreport.com`.

[106] Nor Shahida Mohd Jamail and Rodziah Atan. Service level agreement checking in cloud computing in terms of verification and validation concepts. In *The Third International Conference on Digital Information Processing and Communications*, pages 727–737. The Society of Digital Information and Wireless Communication, 2013. URL `http://sdiwc.net/digital-library/web-admin/upload-pdf/00000475.pdf`.

[107] VS Janakiraman, R Ganesan, and M Gobi. Hybrid cryptographic algorithm for robust network security. In *The International Congress for global Science and Technology*, volume 17, page 33, 2007.

[108] Arshad Jhumka, Felix Freiling, Christof Fetzer, and Neeraj Suri. An approach to synthesise safe systems. *International Journal of Security and Networks*, 1(1-2):62–74, 2006.

[109] Nesrine Kaaniche, Mohamed Mohamed, Maryline Laurent, and Heiko Ludwig. Security sla based monitoring in clouds. In *2017 IEEE International Conference on Edge Computing (EDGE)*, pages 90–97. IEEE, 2017.

[110] S Kahneman and A Tversky. Loss aversion, 1979.

[111] Dalia Khader, Julian Padget, and Martijn Warnier. Reactive monitoring of service level agreements. *Grids and Service-Oriented Architectures for Service Level Agreements*, pages 13–22, 2010. URL `http://www.academia.edu/download/45662121/slaws09.pdf`.

[112] Dalia Khader, Julian Padget, and Martijn Warnier. Reactive monitoring of service level agreements. In *Grids and Service-Oriented Architectures for Service Level Agreements*, pages 13–22. Springer, 2010. URL `http://www.academia.edu/download/45662121/slaws09.pdf`.

[113] Dilshad H Khan and Deepak Kapgate. Efficient virtual machine scheduling in cloud computing. 2014.

[114] Eunjin Ko, Junwoo Lee, Gilhaeng Lee, and Youngsun Kim. The web-based sla monitoring and reporting (WSMR) system. In *ICETE*, pages 293–297. INSTICC Press, 2005.

[115] Charlotte Kotas, Thomas Naughton, and Neena Imam. A comparison of amazon web services and microsoft azure cloud platforms for high performance computing. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4. IEEE, 2018.

[116] Ajay D Kshemkalyani and Mukesh Singhal. *Distributed computing: principles, algorithms, and systems*. Cambridge University Press, 2011.

[117] Prakash Kuppuswamy, Rajan John, et al. A novel approach of designing e-commerce authentication scheme using hybrid cryptography based on simple symmetric key and extended linear block cipher algorithm. In *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, pages 1–6. IEEE, 2020.

[118] Philipp Leitner, Anton Michlmayr, Florian Rosenberg, and Schahram Dustdar. Monitoring, prediction and prevention of sla violations in composite services. In *IEEE Web Services (ICWS) 2010*, pages 369–376. IEEE.

[119] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2013:94, 2012.

[120] Wenjing Lou and Yuguang Fang. A survey of wireless security in mobile ad hoc networks: challenges and available solutions. In *Ad Hoc Wireless Networking*, pages 319–364. Springer, 2004.

[121] Mario Macías and Jordi Guitart. Sla negotiation and enforcement policies for revenue maximization and client classification in cloud providers. *Future Generation Computer Systems*, 41:19–31, 2014.

[122] Zainab Hikmat Mahmood and Mahmood Khalel Ibrahem. New fully homomorphic encryption scheme based on multistage partial homomorphic encryption applied in cloud computing. In *2018 1st Annual International Conference on Information and Sciences (AiCIS)*, pages 182–186. IEEE, 2018.

[123] Dindayal Mahto and Dilip Kumar Yadav. Performance analysis of rsa and elliptic curve cryptography. *IJ Network Security*, 20(4):625–635, 2018.

[124] Ayoub Mars and Wael Adi. Fair exchange and anonymous e-commerce by deploying clone-resistant tokens. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1226–1231. IEEE, 2019.

[125] Mohit Marwaha, Rajeev Bedi, Amritpal Singh, and Tejinder Singh. Comparative analysis of cryptographic algorithms. *Int J Adv Engg Tech/IV/III/July-Sept*, 16:18, 2013.

[126] masterclass. Loss aversion explained: 3 examples of loss aversion. 2020. URL `https://www.masterclass.com/articles/loss-aversion-explained`.

[127] Sjouke Mauw, Sasa Radomirovic, and Mohammad Torabi Dashti. Minimal message complexity of asynchronous multi-party contract signing. In *Computer Security Foundations Symposium, 2009. CSF'09. 22nd IEEE*, pages 13–25. IEEE, 2009. URL `https://pdfs.semanticscholar.org/8a6f/aa47abbb5b5855d4cc8db9437537a9c4d0a9.pdf`.

[128] Patrick McCorry, S. F. Shahandashti, and Feng Hao. A smart contract for boardroom voting and maximum voter privacy. In *21st conference on Financial Cryptography and Data Security*, 2017.

[129] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[130] Daniel A Menascé. Qos issues in web services. *IEEE internet computing*, 6(6):72–75, 2002.

[131] Manuel Humberto Santander Pelaez Michael Hoehl. Proposal for standard cloud computing security slas – key metrics for safeguarding confidential data in the cloud. *SANS Institute Reading Room*.

[132] Microsoft. Sla summary for azure services, 2017. URL `https://azure.microsoft.com/en-gb/support/legal/sla/summary/`.

[133] Microsoft. Azure status history, #mar# 2017. URL `https://azure.microsoft.com/en-us/status/history/`.

[134] Sonam Mittal and KR Ramkumar. Research perspectives on fully homomorphic encryption models for cloud sector. *Journal of Computer Security*, (Preprint):1–26, 2021.

[135] Jeffrey C Mogul and John Wilkes. Nines are not enough: meaningful metrics for clouds. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 136–141, 2019.

[136] Mohamed Mohamed, Obinna Anya, Takashi Sakairi, Samir Tata, NagaPramod Mandagere, and Heiko Ludwig. The rsla framework: Monitoring and enforcement of service level agreements for cloud services. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 625–632. IEEE, 2016.

[137] Sahar Mohamed, Adil Yousif, and Mohammed Bakri. Sla violation detection mechanism for cloud computing. *International Journal of Computer Applications*, 133(6):8–11, 2016. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.742.277&rep=rep1&type=pdf`.

[138] Sahar Mohamed, Adil Yousif, and Mohammed Bakri. Sla violation detection mechanism for cloud computing. *International Journal of Computer Applications*, 133(6):8–11, 2016.

[139] Carlos Molina-Jimenez, Santosh Shrivastava, Jon Crowcroft, and Panos Gevros. On the monitoring of contractual service level agreements. In *Proceedings. First IEEE International Workshop on Electronic Contracting, 2004.*, pages 1–8. IEEE, 2004.

[140] Varsha R Mouli and KP Jevitha. Web services attacks and security-a systematic literature review. *Procedia Computer Science*, 93:870–877, 2016.

[141] Mr. Praveen Kaushik Ms. Shaheen Ayyub. An analysis of security attacks on cloud wrt saas. *International Journal of Advancements in Research & Technology*, Volume 4, Issue, February 2015.

[142] Carlos Muller, Marc Oriol, Xavier Franch, Jordi Marco, Manuel Resinas, Antonio Ruiz-Cortes, and Marc Rodriguez. Comprehensive explanation of sla violations at runtime. *IEEE Transactions on Services Computing*, 7(2): 168–183, 2014. URL `http://idus.us.es/xmlui/bitstream/handle/11441/24591/file_1.pdf?sequence=1&isAllowed=y`.

[143] Snehal Mumbaikar, Puja Padiya, et al. Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, 3(5):1–4, 2013.

[144] Phil Muncaster. Global e-commerce fraud to top usd25bn by 2024. *UK / EMEA News Reporter , Infosecurity Magazine*. URL `https://www.infosecurity-magazine.com/news/global-ecommerce-fraud-to-top-25/`.

[145] Muhammad Faheem Mushtaq, Sapiee Jamel, Abdulkadir Hassan Disina, Zahraddeen A Pindar, N Shafinaz Ahmad Shakir, and Mustafa Mat Deris. A survey on the cryptographic encryption algorithms. *International Journal of Advanced Computer Science and Applications*, 8(11):333–344, 2017.

[146] Falak Nawaz, Omar Hussain, Farookh Khadeer Hussain, Naeem Khalid Janjua, Morteza Saberi, and Elizabeth Chang. Proactive management of

sla violations by capturing relevant external events in a cloud of things environment. *Future Generation Computer Systems*, 95:26–44, 2019.

[147] Minh-Duong Nguyen, Ngoc-Tu Chau, Seungwook Jung, and Souhwan Jung. A demonstration of malicious insider attacks inside cloud iaas vendor. *International Journal of Information and Education Technology*, 4(6):483, 2014.

[148] Mark O'Loughlin. It service management and cloud computing. *Axelos. Acedido*, pages 01–11, 2016.

[149] José Antonio Parejo, Pablo Fernández, Antonio Ruiz Cortés, and José María García. Slaws: Towards a conceptual architecture for sla enforcement. *2008 IEEE Congress on Services - Part I*, pages 322–328, 2008.

[150] Jung-Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Constructing fair-exchange protocols for e-commerce via distributed computation of RSA signatures. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 172–181, 2003. doi: 10.1145/872035.872060. URL `https://doi.org/10.1145/872035.872060`.

[151] Adrian Paschke and Martin Bichler. Sla representation, management and enforcement. In *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 158–163. IEEE, 2005.

[152] Adrian Paschke and Elisabeth Schnappinger-Gerull. A categorization scheme for sla metrics. *Service Oriented Electronic Commerce*, 80(25-40): 14, 2006.

[153] Pankesh Patel, Ajith H Ranabahu, and Amit P Sheth. Service level agreement in cloud computing. 2009.

[154] Priyadarshini Patil, Prashant Narayankar, DG Narayan, and S Md Meena. A comprehensive evaluation of cryptographic algorithms: Des, 3des, aes, rsa and blowfish. *Procedia Computer Science*, 78(1):617–624, 2016.

[155] Kyriacos E Pavlou and Richard T Snodgrass. Temporal implications of database information accountability. In *2012 19th International Symposium on Temporal Representation and Reasoning*, pages 125–132. IEEE, 2012.

[156] Nicole Perlroth and Adam Satariano. Irish hospitals are latest to be hit by ransomware attacks. *The New York Time*, May

2021. URL https://www.nytimes.com/2021/05/20/technology/ransomware-attack-ireland-hospitals.html.

[157] Fabio Piva and Ricardo Dahab. E-commerce and fair exchange-the problem of item validation. In *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pages 317–324. IEEE, 2011. URL http://www.ic.unicamp.br/~reltech/2011/11-05.pdf.

[158] Miroslav Popovic. *Communication protocol engineering*. CRC press, 2018.

[159] Roger S Pressman. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[160] Farrukh Qazi, Arshad Jhumka, and Paul Ezhilchelvan. Towards automated enforcement of cloud sla. In *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*, pages 151–156. ACM, 2017.

[161] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 1981.

[162] Rackspace. Managed hosting terms - en. Technical report, Rackspace. URL https://rackspace.com/en-gb/legal/managed-hosting-terms-uk.

[163] Mussadiq Abdul Rahim, Irfan Ul Haq, Hanif Durad, and Erich Schikuta. Generalized sla enforcement framework using feedback control system. *2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*, pages 1–6, 2015.

[164] Mohammad Saidur Rahman, Ibrahim Khalil, Abdulatif Alabdulatif, and Xun Yi. Privacy preserving service selection using fully homomorphic encryption scheme on untrusted cloud service platform. *Knowledge-Based Systems*, 180:104–115, 2019.

[165] Franco Raimondi, James Skene, and Wolfgang Emmerich. Efficient online monitoring of web-service slas. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 170–180. ACM, 2008.

[166] Noëlle Rakotondravony, Benjamin Taubmann, Waseem Mandarawi, Eva Weishäupl, Peng Xu, Bojan Kolosnjaji, Mykolai Protsenko, Hermann De Meer, and Hans P Reiser. Classifying malware attacks in iaas cloud environments. *Journal of Cloud Computing*, 6(1):26, 2017.

[167] Omer F Rana, Martijn Warnier, Thomas B Quillinan, Frances Brazier, and Dana Cojocarasu. Managing violations in service level agreements. *Grid Middleware and Services*, pages 349–358, 2008.

[168] Johan Rangardt and Matthias Czaja. Empirical investigation of how user experience is affected by response time in a web application. 2017.

[169] Indrajit Ray, Indrakshi Ray, and Narasimhamurthi Natarajan. An anonymous and failure resilient fair-exchange e-commerce protocol. *Decision Support Systems*, 39(3):267–292, 2005.

[170] Register. Aws's s3 outage was so bad amazon couldn't get into its own dashboard to warn the world, March 2017. URL `https://www.theregister.co.uk/2017/03/01/aws_s3_outage/`.

[171] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. *NCM*, 9:44–51, 2009.

[172] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security*, 2009.

[173] Ivan Ristic. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Feisty Duck, 2013.

[174] Ronald L Rivest. The md4 message digest algorithm. In *Conference on the Theory and Application of Cryptography*, pages 303–311. Springer, 1990.

[175] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[176] Sidney Rosario, Albert Benveniste, and Claude Jard. Monitoring probabilistic slas in web service orchestrations. In *Integrated Network Management*, pages 474–481. IEEE, 2009.

[177] Akhil Sahai and Sven Graupner. *Web Services in the Enterprise*, volume 376. Springer, 2005.

[178] Akhil Sahai, Anna Durante, and Vijay Machiraju. Towards automated sla management for web services. *Hewlett-Packard Research Report HPL-2001-310 (R. 1)*, 2002.

[179] Akhil Sahai, Vijay Machiraju, Mehmet Sayal, Aad Van Moorsel, and Fabio Casati. Automated sla monitoring for web services. *Management Technologies for E-Commerce and E-Business Applications*, pages 28–41, 2002.

[180] T Sakthisree, S Kumaresan, D Manisha, and M Prathapkannan. Secure logging as a service in cloud. 2017.

[181] Christian Schubert, Michael Borkowski, and Stefan Schulte. Trustworthy detection and arbitration of sla violations in the cloud. In *ESOCC*, 2018.

[182] Pradeep Semwal and Mahesh Kumar Sharma. Comparative study of different cryptographic algorithms for data security in cloud computing. In *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall)*, pages 1–7. IEEE, 2017.

[183] Z Shams and R Hasan. Cloud forensics: A meta-study of challenges. *Approaches, and Open Problems. arXiv preprint*, 2013.

[184] Esther Shein. Cyberattack on it services giant cognizant impacts clients. April 2020. URL `https://news.cognizant.com/2020-04-18-cognizant-security-update`.

[185] Quan Z Sheng, Xiaoqiang Qiao, Athanasios V Vasilakos, Claudia Szabo, Scott Bourne, and Xiaofei Xu. Web services composition: A decade's overview. *Information Sciences*, 280:218–238, 2014.

[186] Adam Shostack. Experiences threat modeling at microsoft. *MODSEC@ MoDELS*, 2008, 2008.

[187] Adam Shostack. *Threat modeling: Designing for security.* John Wiley & Sons, 2014.

[188] Mohammed Shuaib, Abdus Samad, Shadab Alam, and Shams Tabrez Siddiqui. Why adopting cloud is still a challenge?—a review on issues and challenges for cloud migration. *Ambient Communications and Computer Systems: RACCCS-2018*, 904:387, 2019.

[189] Andrew P Snow and Gary R Weckman. What are the chances an availability sla will be violated? In *Sixth International Conference on Networking (ICN'07)*, pages 35–35. IEEE, 2007.

[190] The Law Society. Cloud computing. August 2018. URL `https://www.lawsociety.org.uk/topics/cybersecurity/cloud-computing`.

[191] Conn STAMFORD. Gartner forecasts worldwide public cloud revenue to grow 6.32020. *Newsroom, Press Releases July*, July 2020.

[192] Statista. E-commerce worldwide - statistics and facts. October 2020. URL https://www.statista.com/topics/871/online-shopping/.

[193] Yusong Tan, Fuhui Wu, Qingbo Wu, and Xiangke Liao. Resource stealing: a resource multiplexing method for mix workloads in cloud system. *The Journal of Supercomputing*, 75:33–49, 2015.

[194] Rajat Tandon, Jelena Mirkovic, and Pithayuth Charnsethikul. Quantifying cloud misbehavior. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–8. IEEE, 2020.

[195] Rana Tassabehji and Ray Hackney. Hey you get off my cloud evaluation of cloud service models for business value within pharma x. *Journal of Advances in Management Sciences & Information Systems*, 2:48–52, 2016.

[196] Tom Tedrick. Fair exchange of secrets. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 434–438. Springer, 1984. ISBN 3-540-15658-5. doi: 10.1007/3-540-39568-7_34. URL https://doi.org/10.1007/3-540-39568-7_34.

[197] Tom Tedrick. How to exchange half a bit. In *Advances in cryptology*, pages 147–151. Springer, 1984.

[198] Douglas Terry, David Goldberg, David Nichols, and Brian Oki. Continuous queries over append-only databases. *Acm Sigmod Record*, 21(2): 321–330, 1992.

[199] trustwave. 2020 trustwave global security report. May 2020. URL https://www.trustwave.com/en-us/resources/library/documents/2020-trustwave-global-security-report/.

[200] Mitch Tulloch. *Microsoft encyclopedia of networking*. Microsoft Press, Redmond, Wash, 2000. ISBN 0735605734.

[201] Rafael Brundo Uriarte, Rocco De Nicola, and Kyriakos Kritikos. Towards distributed sla management with smart contracts and blockchain. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 266–271. IEEE, 2018.

[202] Shahin Vakilinia, Catherine Truchan, James Kempf, and Halima Elbiaze. Automated enforcement of sla for cloud services. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 49–56. IEEE, 2018.

[203] W3. Status code definitions, #jun# 2018. URL `https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`.

[204] Charles X Wang and Scott Webster. The loss-averse newsvendor problem. *Omega*, 37(1):93–105, 2009.

[205] Guilin Wang. An abuse-free fair contract-signing protocol based on the rsa signature. *IEEE Transactions on Information Forensics and Security*, 5(1):158–168, 2009.

[206] Ziqi Wang, Rui Yang, Xiao Fu, Xiaojiang Du, and Bin Luo. A shared memory based cross-vm side channel attacks in iaas cloud. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 181–186. IEEE, 2016.

[207] Amir Teshome Wonjiga, Louis Rilling, and Christine Morin. Security monitoring sla verification in clouds: the case of data integrity. 2019.

[208] Rui Xie and Rose Gamble. A tiered strategy for auditing in the cloud. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 945–946. IEEE, 2012.

[209] Shaoan Xie, Zibin Zheng, Weili Chen, Jiajing Wu, Hong-Ning Dai, and Muhammad Imran. Blockchain for cloud exchange: A survey. *Computers & Electrical Engineering*, 81:106526, 2020.

[210] Jun Xu and Wooyong Lee. Sustaining availability of web services under distributed denial of service attacks. *IEEE Trans. Computers*, 52:195–208, 2003.

[211] Lin Ye, Hongli Zhang, Jiantao Shi, and Xiaojiang Du. Verifying cloud service level agreement. In *(GLOBECOM), 2012 IEEE*, pages 777–782. IEEE, 2012.

[212] Shuai Yuan. *Three Essays on SLA Management in the Availability-Aware Cloud*. PhD thesis, State University of New York at Buffalo, 2018.

[213] Muhammad Sharjeel Zareen, Adeela Waqar, and Baber Aslam. Digital forensics: Latest challenges and response. In *Information Assurance (NCIA), 2013 2nd National Conference on*, pages 21–29. IEEE, 2013.

[214] Hongli Zhang, Lin Ye, Jiantao Shi, Xiaojiang Du, and Mohsen Guizani. Verifying cloud service-level agreement by a third-party auditor. *Security and Communication Networks*, 7(3):492–502, 2014.

[215] Yinqian Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Cross-tenant side-channel attacks in paas clouds. In *Proceedings of*

the 2014 ACM SIGSAC Conference on Computer and Communications Security, pages 990–1003. ACM, 2014.

[216] Huan Zhou, Xue Ouyang, Zhijie Ren, Jinshu Su, Cees de Laat, and Zhiming Zhao. A blockchain based witness model for trustworthy cloud service level agreement enforcement. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1567–1575. IEEE, 2019.

[217] Huan Zhou, Xue Ouyang, Jinshu Su, Cees de Laat, and Zhiming Zhao. Enforcing trustworthy cloud sla with witnesses: A game theory–based model using smart contracts. *Concurrency and Computation: Practice and Experience*, page e5511, 2019.

[218] Jianying Zhou and Dieter Gollman. A fair non-repudiation protocol. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 55–61. IEEE, 1996. URL `http://ai2-s2-pdfs.s3.amazonaws.com/2bb8/dbb13aba067bb5dc1fe813d5f6c5f195f9eb.pdf`.